НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

УДК 004.94

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: «*Комплекс задач прогнозування часових рядів з використанням технології Apache Spark* »

Виконав: студент *4* курсу, групи *ІС-52*
Олійник Ярослав Миколайович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст. викл. Олійник Ю.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *ст.викл. Халус О. А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н., доц. Пасько В.П.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) *6.050101*

«Комп’ютерні науки» («Інформаційні управляючі системи та технології»)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)

“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Олійнику Ярославу Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема проекту « *Комплекс задач прогнозування часових рядів з використанням технології Apache Spark*»

керівник проекту *Олійник Юрій Олександрович, ст. викл.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “*23*”*квітня* 2019 р. №1181-с

2. Термін подання студентом проекту “*03*”*червня* 2019 року

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд аналогів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних

3. Математичне забезпечення: змістовна та математична постановки задачі, алгоритм багаторядного методу групового урахування аргументів

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна діяльності

2. Схема структурна варіантів використань

3. Схема структурна класів програмного забезпечення

4. Схема структурна послідовності

5. Схема структурна компонентів програмного забезпечення

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«15» лютого 2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*19.02.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*21.02.2019*	
3.	*Постановка та формалізація задачі*	*22.02.2019*	
4.	*Розробка інформаційного забезпечення*	*26.02.2019*	
5.	*Алгоритмізація задачі*	*28.02.2019*	
6.	*Обґрунтування використовуваних технічних засобів*	*05.03.2019*	
7.	*Розробка програмного забезпечення*	*08.03.2019*	
8.	*Налагодження програми*	*02.05.2019*	
9.	*Виконання графічних документів*	*10.05.2019*	
10.	*Оформлення пояснювальної записки*	*29.05.2019*	
11.	*Подання ДП на попередній захист*	*30.05.2019*	
12.	*Подання ДП на основний захист*	*03.06.2019*	
13.	*Подання ДП рецензенту*	*05.06.2019*	

Студент _____ Я.М. Олійник
(підпис)

Керівник проекту _____Ю.О. Олійник
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
			Документи текстові		
1	А4	ДП ІС-5220.1181-с.ПЗ	Пояснювальна записка	69	
2	А4	ДП ІС-5220.1181-с.ТЗ	Технічне завдання	10	
3	А4	ДП ІС-5220.1181-с.ПМВ	Програма та методика випробувань	12	
			Матеріали графічні		
4	А3	ДП ІС-5220.1181-с.ССД	Схема структурна діяльності	1	
5	А4	ДП ІС-5220.1181-с.ССВ	Схема структурна варіантів використань	1	
6	А3	ДП ІС-5220.1181-с.ССК	Схема структурна класів програмного	1	
7	А3		забезпечення		
8	А4	ДП ІС-5220.1181-с.ССП	Схема структурна послідовності	1	
9	А4	ДП ІС-5220.1181-с.ССК	Схема структурна компонентів	1	

Зм.	Арк.	ПІП	Підп.	Дата	ДП ІС-5217.1181-с.ВДП		
Розробн.		Олійник Я.М.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Олійник Ю.О.					
Консульт.						КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н/контр.		Халус О.А.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: Комплекс задач прогнозування часових рядів з використанням технології Apache Spark

Київ – 2019 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з шести розділів, містить 104 сторінки, 10 рисунків, 7 таблиць, 1 додаток і 23 джерел.

Дипломний проект присвячений комплексу задач часових рядів із використанням технології Apache Spark.

В розділі загальних положень було обґрунтовано вибір представлених алгоритмів часових рядів та було зазначено яким чином технологія Apache Spark може покращити дані алгоритми.

В розділі інформаційного забезпечення було розглянуто тип даних на вхід та вихід, була описана структура масивів інформації.

В математичному розділі було описано змістовну постановку задачі, були реалізовані алгоритми, а саме багаторядний метод групового урахування аргументів і генетичний алгоритм. Був наданий покроковий опис розглянутих в цій дипломній роботі алгоритмів.

В розділі присвяченому програмному забезпеченню було показано структури додатку, такі як структурна схема класів, розгортання системи, схема діятьності акторів.

В технологічному розділі було описано як користувач мусить взаємодіяти із програмним забезпеченням.

БАГАТОРЯДНИЙ МЕТОД ГРУПОВОГО УРАХУВАННЯ АРГУМЕНТІВ, ЧАСОВІ РЯДИ, ГЕНЕТИЧНИЙ АЛГОРИТМ, ТЕХНОЛОГІЯ APACHE SPARK

ABSTRACT

Structure and scope of work. The explanatory work of the thesis consists of six sections, contains 104 pages, 10 pictures, 7 tables, 1 application, and 23 sources.

The thesis was dedicated to a set of problems of time series with the use of Apache Spark technology.

In the general regulations section, there was described the choice of the following algorithms of time series and there was pointed the way in which the Apache Spark technology could enhance the algorithms.

In the information management section, there was considered the in and out data types, the structure of information datasets was described as well.

In the math section a mission statement was described, the algorithms were implemented, such as a group method of data handling and genetic algorithm. A step-by-step solution of the algorithms was represented as well.

In a section dedicated to the software, it was shown different structures of the application such as the structural scheme of classes, the deployment of the system, scheme, that represents actirs activities.

In the technological section, it was described how the user should interact with

GROUP METHOD OF DATA HANDLING, TIME SETS, GENETIC ALGORITHM, APACHE SPARK TECHNOLOGY

ЗМІСТ

ВСТУП

В сучасному світі все більше популярності набувають технології, які можуть прогнозувати деякий набір даних на майбутнє. Це пов’язано з тим, що саме в цей час починають з’являтися комп’ютери потрібної потужності. Крім того, почали й з’являтися сучасні технології та фреймворки, які застосовуються для швидшого знаходження поставлених задач.

Таким чином, кілька років тому великі компанії, такі як Google, Amazon, Netflix почали впроваджувати алгоритми передбачування у свої системи. Це суттєво вплинуло на якість обслуговування, що в свою чергу призвело до зростання кількості клієнтів. Зрозуміло, що застосування технологій передбачення в даних компаніях не змогло пройти незаміченим, і вже менші компанії почали застосовувати подібні технології у своїх сервісах.

На сьогоднішній день, більшість компаній за кордоном зберігають і використовують Big Data [1] для покращення своїх сервісів. За прогнозами експертів, за кілька років, підприємство, що не матиме вбудованих технологій з прогнозування в свої системи, будуть неконкурентноспроможними. Тому, попит на швидкі, сучасні та якісні алгоритми з прогнозування зростає кожного дня.

Існує досить багато алгоритмів, що дозволяють передбачити часові ряди, такі як: модель Хольга [2], модель ковзаючої середньої та авторегресії [3]. Усі ці алгоритмі досить популярні на заході та в США, але я хотів би реалізувати алгоритм, що був створений українським академіком Олексієм Григоровичем Івахненком: МГУА – метод групового урахування аргументів[4]. Даний метод має багато переваг перед своїми попередниками, одна з яких – це точність прогнозу. Але даний алгоритм не є досить популярним для реалізації бізнес-задач. По-перше, алгоритм не досить швидкий, хоча й швидкість можна варіювати, але якість прогнозу

відповідно, також зменшиться. По-друге, він просто не досить відомий на заході, що я особисто вважаю великим непорозумінням.

Кілька років тому, в 2014 році була випущена нова технолгія Apache Spark[5] для найбільш оптимального автоматичного розподілення ресурсів комп'ютера, що дозволяє значно пришвидшити різні алгоритми, в тому числі і МГУА. Найкраще вона працює із YARN(Yet Another Resource Negotiator)[6], що допомагає розподіляти ресурси кластера. Ця технологія одразу завоювала симпатію великих компаній і вже кілька років використовується в таких компаніях, як IBM, Amazon, Microsoft та інших. Прийшовши на зміну MapReduce[7], Spark змінив уяву про швидкість багатьох алгоритмів. На сьогоднішній день, всі великі підприємства, що раніше використовували MapReduce або MapReduce 2.0 в своїх рішеннях, перейшли на Apache Spark.

Так як, метод групового урахування аргументів є досить точним методом із проблемою в швидкості знаходження оптимальних моделей, то Apache Spark міг вирішити багато проблем, пов'язаних із часом роботи даного методу. Звертаючи увагу на те, що потреба в алгоритмах з точним прогнозуванням наразі лише зростає, на мою думку, реалізуати даний метод на новітніх технологіях стає необхідінстю. Також, необхідно пам'ятати, що якщо ми працюємо із справді масивними даними, то може бути необхідність розгорнути нашу задачу, використовуючи технологію Hadoop, що дасть можливість використовувати усі найновітніші інструменти для роботи з BigData на даний момент. Це, в першу чергу YARN, Pig[8], Hive[9], Sqoop[10] та інші. Тому, навіть після завершення написання дипломного проєкту, можна буде удосконалювати дану роботу і підняти її навіть на вищий рівень.

Практичне значення розроблених результатів. Розроблена система прогнозування часових рядів з використанням технології Apache Spark.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Прогнозування часових рядів на сьогодні не закінчується лише прогнозуванням погодніх явищ, чи передбаченням поведілки валют на ринку, але йде далі. Все більше компаній почали застосовувати дані технології для поліпшення сервісу.

Метод групового урахування аргументів є досить потужним механізмом передбачення різних видів даних. В поєднанні з новою технологією Apache Spark алгоритм починає набувати не тільки точноті прогнозу, але й швидкості.

Так як потужності машин на яких проводять експерименти збільшуються постійно, різні технології прогнозування можуть давати надзвичайно точні прогнози за рахунок охоплення більшої кількості даних, в порівнянні з часами, коли швидкості процесорів були досить сильно обмежені.

Алгоритми, які будуть реалізовані в цьому дипломному проєкті, а саме – багаторядний метод групового урахування аргументів та генетичний алгоритм, є як схожими, так і суттєво відрізняються в плані їхньої розробки. Багаторядний метод групового урахування аргументів є одним із найкращих алгоритмів прогнозування на сьогоднішній день, в той час як генетичний алгоритм представляє собою евристичний метод пошуку оптимального поліному. Зародження генетичного алгоритму почалося саме з багаторядного МГУА, при тому що генетичний алгоритм був створений 20 років після створення МГУА.

За даними на офіційному сайті Apache Spark[5], дана технологія має в 100 разів більшу продуктивність при обробці даних ніж в схожому механізмі MapReduce. ДанаF технологія може використовуватися для розв’язання різних задач штучного інтелекту(машинного навчання).

Spark був розроблений саме для заміни застарілого MapReduce. Дана технологія полегшує розробку ітеративних алгоритмів. Серед нобору

ітеративних алгоритмів, є багато тренуючих алгоритмів для систем навчання штучного інтелекту, які сформували основний стимул для розробки Apache Spark.

Тому в даній дипломній роботі, буде показано новий погляд на метод групового урахування аргументів та генетичний алгоритм, які будуть працювати швидше за будь-які реалізації, зроблені за допомогою інших технологій, таких як MapReduce або розпаралелення вручну.

1.1.1 Опис процесу діяльності

Розглянемо процес діяльності системи. Для відображення процесу діальності користувача, була побудована схема структурна діяльності, що наведена в графічному матеріалі.

Отже, для початку, після завантаження програмного забезпечення на операціюну систему Windows, користувач заходить в програму.

В даній системі буде присутній лише один актор – користувач. Так як представлене програмне забезпечення розроблялося з метою проведення дослідження роботи новінтіх технологій керування ресурсами на алгоритмі МГУА та генетичному алгоритмі, то інших акторів вона не потребує.

Після запуску програмного забезпечення користувачеві буде запропоновано створити новий прогноз.

Під час створення нового прогнозу, користувач може брати алгоритм, за допомогою якого буде отримувати результати.

Користувачеві буде запропоновано спосіб введення даних:

- введення даних вручну;
- завантаження даних із жорсткого диску.

По тому, як програма отримала дані, потрібно виставити параметри для нашої задачі:

- задання початку діапазону коефіцієнтів;
- задання кінця діапазону коефіцієнтів;

- задання кроку зміни коефіцієнтів;
- задання кількості нових моделей, що мають генеруватися на наступних кроках;
- задання точності критерію;
- задання пропорції, в якій буде вказано скільки відсотків даних відведеться на тренування моделі, а скільки на побудову;

Після введення усіх потрібних параметрів, можна запускати модель для знаходження прогнозу.

Робота алгоритму може зайняти деякий час, залежно від діапазону коефіцієнтів, що були задані, кроку зміни, кількості нових моделей та точності критерію.

Після знаходження моделі, програма видасть графік, що буде відображати дані, що були введені та прогнозовані дані. Також, в текстовому режимі буде вказаний поліном оптимальної моделі, що був знайдений разом із поліномом для кожного аргументу, що був виражений через інші аргументи.

1.1.2 Опис функціональної моделі

Актором системи є лице Користувач.

Всі дії або варіанти використання акторів, наведені в таблиці 1.1. В даній талиці описаний актор, варіанти використання та опис його дій. Дії або варіанти використання, що виконують в системі актори, наведені в таблиці 1.1, в якій описані актори, варіанти використання та їх описи дій.

Таблиця 1.1– Типи залежностей між варіантами використання

Актор	*Варіант використа-ння*	*Опис дії варіанта використання*
Користувач	Створення нового прогнозу	Користувач може створити новий прогноз після натиску на клавішу «Створити».
	Огляд прогнозу	Користувач може оглянути створений прогноз.
	Обирання алгоритму	Користувач може обрати алгоритм, за яким буде здійснюватися пошук прогнозу
	Додавання даних з жорсткого диску	Користувач відкриває файл з підготовленими даними для прогнозування за допомогою клавіші «Відкрити».
	Додавання даних вручну	Користувач починає вводити дані власноруч, якщо в нього не було підготовленого файлу з даними.
	Задання початку діапазону коефіцієнтів	Користувач задає початковий коефіцієнт, від якого модель буде підбирати коефіцієнти біля змінних.

Продовження таблиці 1.1

Актор	*Варіант використа-ння*	*Опис дії варіанта використання*
	Задання кінця діапазону коефіцієнтів	Користувач задає кінцевий коефіцієнт, до якого модель буде пыдбирати коефіцієнти біля змінних.
	Задання кількості нових моделей	Користувач може варіювати кількістю нових моделей, що будуть створені на новому кроці.
	Задання точності критерію	Користувач задає точність критерію. Чим меншим буде критерій, тим більш точніший прогноз буде отримано.
	Задання пропорції тренувальних даних та даних для перевірки	Користувач обирає відсоток даних, які підуть на тренування(навчання) моделі та кількість даних, на яких буде перевірятись якість прогнозу.
	Отримання графічного відобреження результату	Користувач отримує графік із введеними ним даними та прогнозованими даними.

Продовження таблиці 1.1

Актор	*Варіант використа-ння*	*Опис дії варіанта використання*
	Отримання текстового відображе-ння результату	Користувач отримує дані прогнозу, виведені в текстовому форматі.

Відповідно до зазначених вище варіантів використання, для відображення моделі використання користувача, була побудована структурна схема використання, що наведена в графічному матеріалі.

1.2 Огляд наявних аналогів

Так як все більше компаній хочуть залучати алгоритми передбачення, то з’являється попит на програми даного типу. Відповідно, попит заповнюється пропозицією, яка є на ринку.

Наразі існує недостатня кількість програмних забезпечень, що могли б бути вендорами з надання високоякісних прогнозів. Саме тому, ціни на відповідні продукти коливаються в районі 1000$ в місяць. Через відсутність альтернативи, фірми змушені переплачувати, щоб отримувати нових клієнтів та покращувати сервіс.

Потрібно також розуміти, що не завжди прогнозування використовується лише в бізнесі. Також подібні рішення можна знайти і в сфері медицини або безпеки. Найчастіше в цій дипломній роботі буде розглядатися порівняння із програмами, що допомагають саме бізнесу, тому що в інших випадках досить важко отримати доступ до програмних продуктів пов’язаних із медициною або національною безпекою, так як такі програмні

продукти знаходяться на локальних машинах, і їх неможливо скачати через інтернет.

Алгоритми передбачення часових рядів зробили також великий внесок і в медицину і національну безпеку. Активно розробкою алгоритмів, які в майбутньому допоможуть нам передбачувати хвороби займається компанія IBM та низка інших компаній на державне замовлення США.

Один із таких алгоритмів дозволяє передбачити в якому регіоні Сполучених Штатів Америки розпочнеться епідемія грипу. Алгоритм працює наступним чином: досліджуються запити людей із усіх штатів країни, і якщо в якомусь зі штатів кількість запитів в пошукові системи із симптомами хвороби збільшується, відповідні органи можуть обмежити даний штат, таким чином попередити епідемію грипу.

Інший, не менш цікавий алгоритм дозволяє передбачити прогресування хвороби Альцгеймера навіть до появи будь-яких видимих симптомів. IBM пропонує датчик, що кріпиться на палець та зчитує з якою силою людина підбирає, зжимає, скручує речі. Отримані дані датчик передає на суперкомп’ютер IBM, який після розрахунків передбачення може сказати яка в людини ймовірність занедужати даною хворобою.

Також існує багато засекречених програм із прогнозування часових рядів, які використовуються здебільшого в воєнних цілях, або в цілях державної безпеки. Так як до них немає доступу, досить важко судити які алгоритми використовуються в даних програмних продуктах, і наскільки точні результати отримуються.

Загалом, оглянувши по основні області застосування алгоритмів часових рядів, стає зрозуміло, що найбільш поширеними і доступними є саме комерційні програмні забезпечення, що створені для приватних компаній. Розглянемо найпопулярніші з них:

GoodData [11] одна з найбільших компаній, які займаються аналітикою та розробкою стратегій для бізнесу. Один з основних алгоритмів, яким користується компанія є саме алгоритм МГУА.

В основному клієнтами компанії GoodData є великі корпорації, які ставлять собі за ціль збільшення доходу. Програмне забезпечення досить схоже на те, що описано в даній дипломній роботі, а саме: клієнт вводить дані, обирає час, точність і кількість прогнозованих точок та отримує результат в .xls файл.

Ця компанія вважає, що до 2020 року більш ніж 85% роздрібної торгівлі в інтернеті буде відбуватися за рахунок штучного інтелекту. Компанія співпрацювала із такими світовими брендами як: NextWorld, Windgrest, Intel та інші.

Великим плюсом даної компанії є те, що вона працює зі всіма видами даних, таких як:

- структуровані дані(реляційні таблиці, *.xls – документи);
- напівструктуровані дані(JSON-файли, *.csv – файли);
- неструктуровані дані(текстові документи, веб-сторінки, тощо);

Тобто, користувачеві не потрібно хвилюватися про структуризацію даних під час роботи із програмним забезпеченням.

Компанія реалізовує свій продукт на наступних мовах програмування: Python[12], Ruby[13], R[14], Scala[15]. При цьому перевага йде саме мові Python, так як вона вважається однією із найкращих мов програмування для прогнозування, побудови нейронних мереж та штучного інтелекту.

Також слід зазначити, що дана компанія має дата-центри по всьому світу, зокрема в США, Європі, Канаді, що одночасно пришвидшує отримання результату для клієнта і робить користування сервісу більш комфортним. Також, дані надійно захищені, так як копії даних знаходяться в різних кутках світу. Тобто, при втраті серверів, наприклад, в США, клієнт не втрачає дані, а користується серверами з Європи.

Досить популярною в останніх кілька років стала приватність даних. Для багатьох покупців, особливо великих компаній саме надійність їхніх даних стає ключовим приорітетом при виборі вендора для прогнозування. Дана компанія може похизуватися наступними стандартами до надійності даних: SOC 2, HIPAA, GDPR і ISO 27001.

Загалом, сервіс даної компанії виглядає досить потужним, тому для великих компаній це буде хороший вибір. Але для малих підприємств даний вендор не підходить, так як вирішує набагато ширший об'єм задач, ніж лише прогнозування. Тому співпрацювати і витрачати час для невеликих компаній даний вендор не буде. Також є проблема із підтримкою програмного забезпечення, яка триває лише місяць. Тобто, компанія GoodData розраховує на те, що в компанії-замовника в штаті працівників є аналітик, якого швидко можна навчити працювати із програмним забезпеченням. Але не для всіх компаній це так.

GMDH Streamline [16] (з англ. Group Methods of Data Handling – метод групового урахування аргументів) – Заснована в 2009 році, GMDH Streamline є провідним міжнародним постачальником рішень для прогнозування бізнесу. Рішення GMDH обробляють кожну частину процесу планування попиту та запасів. GMDH Software створює розширені програмні рішення, які приносять потужність алгоритмів моделювання та прогнозування GMDH, забезпечуючи точне, гнучке прогнозування для бізнесу. GMDH Streamline - це цілеспрямоване планування попиту та рішення для управління запасами, що дозволяє компаніям максимізувати віддачу від капітальних інвестицій.

Як зрозуміло з назви, дана компанія основана саме на алгоритмі групового урахування аргументів. Даний вендор є більш спрямованим саме на прогнозування часових рядів, на відміну від компанії вище, тому порівняння з ним буде більш доцільним. Так як GMDH Streamline була заснована у 2009 році, і працює за однієї схемою, то технологія Apache Spark скоріше за все не

була впроваджена в систему. Тобто, софт даної компанії можна було б пришвидшити.

Оглянувши найпопулярніших вендорів із поставки прогнозування часових рядів, можна зробити такі основні висновки:

- програми працюють недостатньо швидко;
- програми в основному націлені на великий бізнес(важко знайти програму з прогнозування не для бізнесу);
- ціна використання таких програм – близько 1000-2000$ на місяць;

В даному дипломному проєкті будуть вирішені вищенаведені недоліки.

Також слід розглянути існуючі аналоги бібліотек, за допомогою яких можна розподіляти ресурси комп'ютера для пришвидшення роботи алгоритмів.

Так як багатоядерні вичислювальні машини вже 15 років використовуються повсякденно, то було розроблено достатню кількість бібліотек, які могли б використовувати усі ресурси комп'ютера.

Можна виділити кілька найпопулярніших бібліотек для розпаралелення:

MPI [17] – розшифровується як "Message passing interface" ("Взаємодія через передачу повідомлень. MPI дає програмісту єдиний інтерфейс взаємодії гілок всередині паралельного додатку незалежно від машинної архітектури, взаємного розташування гілок і API операційної системи.

Наразі різними колективами розробників було написано декілька програмних пакетів, що задовольняють специфікації MPI, а саме: MPICH, LAM, HPVM і так далі. Вони є базовими при переносі MPI на нові архітектури.

Будь-яка прикладна MPI-програма повинна починатися з виклику функції ініціалізації MPI: функції MPI_Init. В результаті виконання цієї функції створюється група процесів, в яку поміщаються всі процеси додатку, і створюється область зв'язку, що описується перевизначеним комунікатором MPI_COMM_WORLD. Ця область зв'язку об'єднує всі процеси. Процеси в

групі впорядковані від 0 до size-1, де size рівне числу процесів в групі. Таким чином відбувається комунікація між процесами.

Недоліком цієї технології є її нетривіальне встановлення на персональний комп'ютер на будь-яке середовище розробки. Бібліотека MPI може бути використана на наступних мовах програмування: C++, Java, Python. Залежно від того, яку мову програмування обрати, відповідно і буде мінятися швидкість. Іншим недоліком можна відмітити те, що користувачеві самостійно потрібно з'ясовувати оптимальну кількість процесів, на яку він буде ділити програму. До того ж, після вибору певної кількості процесів, ця кількість залишається статичною до самого завершення програми, без можливості динамічно змінити кількість процесів.

iPyParallel [18] – бібліотека для розпаралелення програм на мові програмування Python. Як і в MPI в цій бібліотеці існує багато базових команд, що були використані для паралельного програмування.

Існує кілька спільних архітектур, що поєднують обчислення ресунсів для паралельних процесів, і кожна архітектура має різний протокол для розподілення пам'яті і процесорів між нодами. Кожна архітектура пропонує свої унікальні переваги і недоліки, але спільні команди для використання в таких програмах спільні.

Дана бібліотека працює за наступним принципом: спочатку клієнт пише програму, використовуючи дану бібліотеку та запускає її. Контролер отримує шляхи від клієнта і поширює інструкції і дані на ноди. Контролер керує комунікаціями і планувальниками, щоб присвоїти процеси до машини.

Недоліками даної бібліотеки можна вважати те, що для того, щоб використовувати її, необхідно працювати виключно із мовою програмування Python. Також, спільним недоліком в двох представлених бібліотек можна вважати ручне розподілення ресурсів. Клієнту потрібно зробити певні дослідження перед тим, як писати програму, для того, щоб розуміти яка кількість процесів буде оптимальною.

1.3 Постановка задачі

1.3.1 Призначення розробки

Так як алгоритми прогнозування числових рядів набирають все більшої популярності серед великих корпорацій, то попит на прогномне забезпечення зростає постійно. Слід відмітити, що часто алгоритми, що надаються вендорами не застосовують найновітніших технологій в галузі комп'ютерних наук.

Даний дипломний проєкт може бути використавний корпораціями для отримання прогнозів на бізнес-рішення, який буде не лише дешевшим, але й значно швидшим за рахунок використання новітньої технології розподілення ресурсів Apache Spark.

В майбутньому, із даною роботою можна звертатися до інвесторів, щоб пояснити поточну ситуацію на ринку, таким чином отримавши інвестиції в розширення цієї ідеї.

Також призначенням розробки є

- привернення уваги до методу групового урахування аргументів, створеного академіком Івахненком, та популяризації його серед людей, які в майбутньому працюватимуть в Data Science;
- дослідження взаємодії програмного забезпечення Apache Spark із даним алгоритмом.

1.3.2 Цілі та задачі розробки

Цілями розробки системи є:

- прискорення процесу прогнозування даних;
- розширення аудиторії викоритсання програм для прогнозування;
- зменшення ціни на програмні продукти прогнозування;

Для досягнення поставлених цілей необхідно реалізувати наступні задачі.

Задачами розробки є:

- зробити простий та зрозумілий інтерфейс.
- створити набір інструментів, які будуть необхідні для прогнозування великого спектру задач(біржа, прогноз погоди, кількість посіву на наступний рік, кількість закупки матеріалів на наступний рік, тощо);
- пришвидшити роботу алгоритму багаторядного МГУА за допомогою використання технології Apache Spark;
- зробити продукт доступним для широкої аудиторії користувачів.

Із реалізацією задач буде створено зручний інструментарій для створення пронозування різних систем.

Висновок до розділу

У даному розділі здійснений детальний аналіз предметної області. Оглянуто та проаналізовано існуючі аналоги. Визначено цілі задачі розробки.

Також було сформовано призначення розробки, цілі та задачі, які ставилися перед розробкою даного продукту. Було детально оглянуто конкурентів на ринку та описано як в чому даний програмний продукт перевершує конкурентів, так і чого бракує.

Було детально описано дії актора – користувача, який взяємодіє із програмою. Взаємодії були представлені в вигляді таблиці.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Під час проєктування даного продукту, було зроблено висновки, що найоптимальнішим для користувача і розробника подавати дані досліджень у *.csv форматі, так як кількість змінних для часових рядів може варіюватися від двох до кількох сотень. Тому реляційні бази даних не будуть оптимальним рішенням.

Дані, що надходять від користувача:

- дані спостережень;

Дані, що надходять при введенні параметрів для запуску роботи алгоритму:

- вид алгоритму;
- початкова модель;
- багаторядна модель;
- критерій;
- кількість моделей;
- мінімальна границя коефіцієнтів;
- максимальна границя коефіцієнтів;
- крок зміни коефіцієнтів;
- кількість прогнозованих точок.

2.2 Вихідні дані

На вихід користувач отримує графік, що відображує положення введених та прогнозованих даних.

Також користувачу надається текстовий звіт із отриманим прогнозом.

2.3 Структура масивів інформації

Так як програмне забезпечення створюється з метою дослідження роботи технології Apache Spark за допомогою одного із алгоритмів, що дозволяють знайти прогнозування часових рядів, то достатньо мати одного актора, який на вхід буде віддавати свої дані дослідження, а на вихід отримуватиме дані прогнозу.

Тому, в цьому випадку, замість бази даних буде реалізовано механізм запису в *.csv-файл.

Кожне окреме дослідження мало окремий рядок. Дані в рядку розділялися делімітером табуляція. Перше значення дослідження в кожному рядку – значення, яке потрібно спрогнозувати, всі інші – агрументи спостереження.

Висновок до розділу

У даному розділі було розглянуто вхідні та вихідні дані програмного забезпечення. Було обґрунтовано вибір *.csv – файлу, як структуру масивів інофрмації, подану на вхід.

Було детально описано дані, що надходять при введенні параметрів для запуску роботи алгоритму.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Багаторядний алгоритм МГУА історично був створений академіком інституту кібернетики НАНУ Олексієм Григоровичем Івахненком в 1968 році.

Основні ідеї алгоритму:

1) зменшити кількість моделей, що розглядаються на кожному кроці
2) зменшити кількість рядів, і тим самим пришвидшити вихід на оптимальний рівень складності

Тому на кожному ряду:

1) відбирається фіксоване число найкращих моделей(кожна модель розглядається, як змінна)
2) кожна пара найкращих змінних породжує нову змінну при переході на наступний рівень

Даний алгоритм несе в собі ідею генетичних алгоритмів і ідею поліноміальних нейронних мереж. При тому, що він з’явився на 10-20 років раніше.

Ми будемо вибирати з кожного ряду по 5 найкращих моделей до тих пір поки заданий критерій буде зменшуватись із кожним рядом. Як тільки зменшення припинеться, програма обере найкращу поточну модель.

На рисунку 2.2 схематично зображено як має виглядати результат прогнозованих даних. Тут чорні точки – дані за умовою, сині точки – прогнозовані дані.

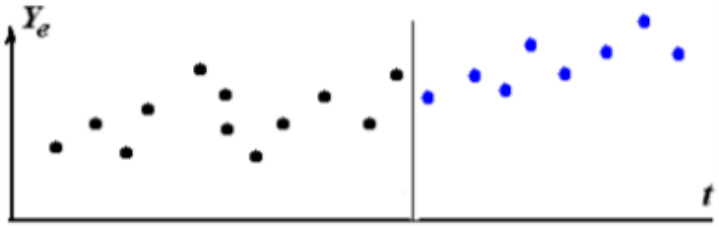


Рисунок 2.2 – Задані та прогнозовані дані

Оскільки чутливість моделі повинна перевірятися на нових даних, то потрібно розбити всю множину даних на 2 частини.

Перший набір даних буде використовуватися для побудови моделі. Цей набір даних буде називатися даними для навчання (Training data).

Другий набір даних буде грати роль нових даних і він буде використовуватися для перевірки якості побудованої моделі. Такий набір даних буде називатися даними для перевірки (Checking data).

Для реалізації ідеї потріно мати кртерій, який би оцінював якість моделі на нових даних. Такий критерій називається критерієм регулярності. Він є зовнішнім критерієм якості моделі.

Оскільки незалежність від даних повнинна перевірятися побудовою моделі на різних даних, то розіб'ємо всю множину на дві частини.

Критерій регулярності[19] може виглядати наступним чином:

$$\sum_C (Y_e - Y_m(T))^2 \qquad (3.1)$$

Y_e – дані спостережень;

$Y_m(T)$ – значення моделі, що була навчена на наборі Training;

Σ_c –сумування по точках набору Checking

Також можна пронормувати критерій, поділишви вираз на $\sum_C (Y_e)^2$.

Багаторядний метод групового урахуваня аргументів, який реалізований в даній дипломній роботі іде далі від комбінаторного алгоритму МГУА. Але також є багато спільних особливостей, які потрібно виділити.

Для того, щоб забезпечити хорошу прогнозну властивість, необхідно мати стійкість до нових даних [20].

Оскільки чутливість моделі повинна перевірятися на нових даних, то потрібно розбити множину значень на дві частини. Перший набір даних буде використовуватися для побудови моделі. Цей набір даних буде називатися даними для навчання(training). Другий набір даних буде грати роль нових даних і він буде використовуватися для перевірки якості побудови моделі. Цей набір даних називається даними для перевірки(checking).

Також, для реалізації ідеї потбірно мати критерій, по якому обирати найкращі моделі. Такий приктерій називається критерієм регулярності. Він є зовнішнім критерієм якості моделі. Ми повинні забезпечити хорошу описову властивість даних, тобто незалежність опису моделі від даних.

Критерій регулярності оцінює стійкість до нових даних. Але також нам потрібно, щоб була забезпечена незалежність моделі від даних. Це можна отримати за допомогою критерія незалежності.

Для оцінки якості потрібно мати ще один незалежний набір даних, так звану екзаменаційну вибірку, verifying data. Саме на цій вибірці перевіряється якість моделі.

Кроки алгоритму з критерієм регулярності:

- визначити серію моделей зростаючої складності;
- експериментальні дані = набір даних для навчання + набір даних для перевірки;
- для заданого рівня складності визначається оцінка параметрів моделі на першому наборі даних. Для цього використовується внутрішній критерій. В якості внутрішнього критерію можна використовувати середньо-квадратичне відхилення справжніх даних від отриманий під час навчання;

- отримана за внутрішнім критерієм модель перевіряється на другому наборі даних(Checking). Використовується зовнішній критерій – критерій регулярності;
- якщо зовнішній критерій досягає мінімуму, то можемо завершувати, інакше, збільшуємо складність моделі і повертаємось на крок 3.

Генетичний алгоритм є еврестичним алгоритмом, який обирає на кожному поколінні найкращі моделі(нащадки). Після обрання певної кількості нащадків, відбувається їхнє схрещування між собою з метою отримання кращого покоління. Схрещування відбувається наступним чином:

- деяким способом обираються 2 моделі-нащадки;
- визначається коефіцієнт схрежування – відношення кількості «генів» від першої моделі до другої;
- відбувається «злиття» двох нащадків, які дають двох потомків;
- дане «злиття» відбувається для всіх моделей нащадків.

Для того, щоб покоління не корелювали між собою, вводиться коефіцієнт мутації. Коефіцієнт мутації визначає скільки відсотків генів від нового покоління будуть мутувати, тобто зміняться.

Зазвичай, обирають невеликий коефіцієнт мутації в рамках 5-10%, щоб нащадки не сильно відрізнялися від батьків.

Після мутації, за допомогою фітнес-функції відбираються найкращі нащадки вже нового покоління.

Алгоритм завершиться після того, як нове покоління дасть гірші результати за старе покоління. Таким чином, щоб перевірити цю умову, після повного формування нового покоління, нам потрібно зберігати дані про старе покоління, щоб ми змогли порівняти фітнес-функції і визначити чи нове покоління краще за старе.

Після отримання найкращого покоління, із нього обирається один нащадок із найкращою фітнес-функцією. Це і буде наш результат прогнозування.

3.2 Математична постановка задачі

Маємо дані спостережень $Y_e(x)$

Потрібно побудувати модель для їх опису

$$Y_m = F(a_1, a_2, \ldots . x) \tag{3.2}$$

Тут $(a_1, a_2, \ldots)$ - параметри моделі

F- структура моделі.

Нехай наша модель – поліном:

$$Y_m = a_0 + a_1 \tag{3.3}$$

$$Y_m = a_0 + a_1 x + a_2 x^2 \tag{3.4}$$

$$Y_m = a_0 + a_3 x^3 \tag{3.5}$$

і так далі.

Рисунок 2.1 - Відображення прогнозованої функції та даних спостережень

Нам задана модель. Потрібно знайти коефіцієнти $(a_1, a_2, \ldots)$ - параметри моделі, при яких критерій буде мінімальний

В загальному, побудова моделей за експериментальними даними може бути записана, як пошук мінімального критерію *CRIT* розглянутий на множині різних моделей ∂:

$$f^* = \arg\min_{f \in \partial} CRIT(f) \tag{3.6}$$

Але (2.1) не є повним формулюванням задачі.

Задана регресійна модель:

$$Y_m = a_0 + a_1 t \tag{3.7}$$

або

$$Y_m = a_0 + a_1 t + a_2 t^2 \tag{3.8}$$

або будь-яка інша. Модель може задаватися користувачем.

Потрібно знайти параметри $(a_1, a_2, \dots)$ з умови:

$$\sum (Y_e - Y_m)^2 \to min \tag{3.9}$$

Ми маємо забезпечити добру прогнозну властивість моделі, тобто стійкість до нових даних.

Розглянемо математичну реалізацію багаторядного алгоритму МГУА.

Нехай число найкращих моделей на кожному ряді – p. Наприклад, нехай p=5. Функція перетворення пари змінних t,s одного ряду в змінні наступного ряду. Приклади можуть бути наступними:

$$Z(t,s) = a_0 + a_t t + a_s s \tag{3.10}$$

$$Z(t,s) = a_0 + a_t t + a_s s + a_{ts} t * s \tag{3.11}$$

$$Z(t,s) = a_0 + a_t t + a_s s + a_{tt} t^2 + a_{ss} s^2 \tag{3.12}$$

Можна використовувати різні функції. В реальних задачах потрібно буде підбирати оптимальний вигляд функції.

Число найкращих моделей і функцію перетворення задає користувач. На першому ряді маємо:

$Y_1^{(1)} = a_0 + a_1 x_1$ $Y_1^{(2)} = a_0 + a_2 x_2 \dots$ $Y_1^{(20)} = a_0 + a_{20} x_{20}$

Обираємо 5 найкращих моделей(так як p=5), нехай

$Y_1^{(2)}, Y_1^{(5)}, Y_1^{(7)}, Y_1^{(13)}, Y_1^{(20)}$.

Ці моделі розглядаються як змінні. На другомі ряді розглядаються функції від всіх пар обраних змінних $Y_1^{(2)}, Y_1^{(5)}, Y_1^{(7)}, Y_1^{(13)}, Y_1^{(20)}$ між собою(тобто моделі виду $Z(Y_1^{(i)}, Y_1^{(j)})$.

Приклад k-ї змінної другого ряду:

$$Z(Y_1^{(i)}, Y_1^{(j)}) = a_{0,k} + a_{2,k} Y_1^{(2)} + a_{7,k} Y_1^{(7)} + a_{2,7,k} Y_1^{(2)} Y_1^{(7)} \tag{3.13}$$

Число змінних M в кожному ряді: M=$C_p^2 = C_5^2 = 10$

3.3 Обґрунтування методу розв’язання

Багаторядний алгорим МГУА є ідеальною комбінацією мінімального набору моделей та найбільш точного результату. Так як пошук коефіцієнтів кожної окремої моделі займає досить велику кількість часу, використовувати алгоритм із повним перебором моделей буде не тільки не доцільно, але й неможливо для прикладних завданя, де використовується більше ніж 20 наборів даних.

Також, потрібно пам’ятати, що даний алгоритм справляється із малою вибіркою даних. Він працює коректно навіть коли кількість даних менша за кількість змінних, видаючи досить непогані результати.

Так як одною із основних цілей даної дипломної роботи є представлення потужності засобу Apache Spark, в якому використовується розпаралелення задачі, даний алгоритм є найкращим для демонстрації роботи засобу компанії Apache.

Сам продукт Apache Spark, за допомогою якого реалізований даний алгоритм, не дає доступу користувачеві до дії розпаралелення. Apache Spark збирає дані про ресурси машини, на якій буде виконуватися алгоритм та самостійно розподіляє ці резурси на процеси, таким чином, щоб це було максимально ефективно. В цьому і відмінність даного застосування від розпаралелення вручну, яке не завжди є максимально ефективним, до того ж вимагає багато часу на розробку та моделювання системи.

Більш того, цей програмний засіб може вдихнути нове життя в багаторядний алгоритм МГУА, що в подальшому може сказатися на ще більший розвиток даного методу і штучного інтелекту вцілому.

3.4 Порівняння методів моделювання

Комбінаторний(однорядний) алгоритм МГУА

Комбінаторний алгоритм вважається найпростішим із всіх можливих алгоритмів групового урахування аргументів. Ідея самого алгоритму: не пропустити жодної з можливих моделей. Тому, на кожному рівні складності розглядаються всі моделі та не проводиться селекція найкращих комбінацій змінних. Таким чином отримуємо, що якщо число змінних моделі N, то число всіх комбінацій M=2^N.

Комбінаторний алгоритм МГУА робить аналіз моделей, що були побудовані за допомогою всіх можливих комбінацій змінних, що були подані на вхід. Тобто, спочатку алгоритм генерує моделі, після чого обчислює їх критерії. Селективним методом, алгоритм обирає модель із найкращим критерієм.

Отримуємо алгоритм, який не пропускає ніодної моделі. Таким чином, як було сказано вище, маємо експоненційне зростання кількості моделей на кожному рівні. Зрозуміло, що це нам не дає змоги використовувати даний алгоритм на великих наборах даних. Тому для бізнес-рішень даний метод не підходить, бо в справжніх системах кількість змінних може досягати кількох сотень.

Експоненційний ріст цього алгоритму обмежує число змінних до N≤ 20.

Робота алгоритму:

На першому ряді використовуємо всі змінні x_i

$Y_1 = a_0 + a_1x_1$ $Y_1 = a_0 + a_2x_2 \dots\dots\dots$ $Y_1 = a_0 + a_{20}x_{20}$

В другому ряді використовуємо всі комбінації з 2-х змінних (x_i, x_j)

$Y_2 = a_0 + a_1x_1 + a_2x_2$ $Y_2 = a_0 + a_1x_1 + a_3x_3 \dots.$ $Y_2 = a_0 + a_{19}x_{19} + a_{20}x_{20}$

В третьому ряді будуть всі комбінації з 3-х змінних (x_i, x_j, x_k)

І так далі.

Число моделей M на кожному ряду для випадку N=20

(число змінних):

Ряд 0: $M_0 = C_{20}^0 = 1$

Ряд 1: $M_1 = C_{20}^1 = 20$

Ряд 2: $M_2 = C_{20}^2 = 190$

Ряд 3: $M_3 = C_{20}^3 = 1140$

Ряд k: $M_k = C_{20}^k$

.....

Ряд N: $M_N = C_{20}^{20} = 1$

Всього моделей(число всіх можливих комбінацій) $M_\varepsilon = \sum_k C_N^k = 2^N$

При N=20 маємо $M_\varepsilon = 2^{20} \approx 100000$

Отримали 1000 000 моделей. Для того, щоб підібрати коефіцієнти до кожної моделі знадобиться досить потужний комп'ютер.

Для того, щоб алгоритм працював швидше під час формування системи, ми можемо використати те, що в комбінаторному алгоритмі достатньо лише перший раз обчислити повну нормальну систему, яка в свою чергу містить всі елементи часткових нормальних систем. Після чого, ми можемо використовувати дану систему для подальших обчислень.

Комбінаторно-селекційний алгоритм МГУА

Основна ідея цього алгоритму полягає в тому, щоб зменшити кількість моделей, що розглядаються на кожному ряді, при цьому по можливості не втратити найкращу комбінацію змінних.

Тому, на кожному рівні складності:

- відбирається фіксоване число найкращих поєднань змінних моделі;
- це найкраще поєднання комбінується зі всіма іншими змінними(по черзі) при переході на наступний рівень.

Нехай параметр алгоритму p=2 – число найкращих моделей в одному ряді.

На першому кроці розглядаємо всі моделі:

$Y_1 = a_0 + a_1 x_1$ $Y_1 = a_0 + a_2 x_2 \ldots\ldots\ldots$ $Y_1 = a_0 + a_{20} x_{20}$

З них обираємо змінні двох найкращих моделей(так як p=2). Нехай це будуть x_3 і x_7.

В другому ряді ці змінні комбінуються зі всіма іншими змінними, тобто маємо наступні пари: $(x_3, x_k), (x_7, x_k)$, де k – будь яка модель.

Отримали:

$$Y_2 = a_0 + a_3x_3 + a_1x_1 \quad (3.14)$$

$$Y_2 = a_0 + a_3x_3 + a_2x_2 \quad (3.15)$$

… … …

$$Y_2 = a_0 + a_7x_7 + a_Nx_N \quad (3.16)$$

З них обираємо змінні двох найкраших моделей(так як p=2). Нехай вони будуть $(x_3, x_2), (x_7, x_{15})$.

Знову повторюємо комбінацію до тих пір, поки критерій буде зменшуватись.

В порівнянні з багаторядним алгоритмом МГУА, де ми будемо отримувати лише 10 нових моделей на кожному новому ряді, вибір алгоритму очевидний.

Таким чином кількість моделей буде значно меншою ніж в випадку із комбінаторним та комбінаторно-селекційним алгортмом МГУА. Саме тому, в даній дипломній роботі буде реалізований саме цей алгоритм.

Висновок до розділу

В даному розділі було розглянуто основні причини, чому на виконання цієї дипломної роботи буде застосовуватися саме багаторядний алгоритм МГУА. Було детально розписано як будуть розподілятися дані для тренування та перевірки моделей, також був представлений алгоритм роботи програми, так само як і алгоритм більш слабкого схожого методу для обґрунтування вибору даного алгоритму, в порівнянні з іншими.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Програмне забезпечення, що використовується при розробці:

- платформа: **OS Windows 10 –** найпопулярніша операційна система, що з легкістю дозволяє використовувати простий та зрозумілий графічний інтерфейс для взаємодії із ресурсами комп'ютеру;
- середовище розробки: **Intellij IDEA –** серія продуктів фірми JetBrains, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом.
- мова написання коду програми: **Java -** об'єктно-орієнтована мова програмування з безпечною системою типізації. Працює більш ніж на 3 мільярдах девайсів**;**
- текстовий редактор для зчитування і запису інформації: **Notepad++ -** зручний редактор із зрозумілим інтерфейсом;
- механізм для роботи з розпаралеленням системи: **Apache Spark –** новітній набір фреймворків, що дозволяє значно пришвидшувати деякі алгоритми. Може працювати в поєднанні з різними мовами програмування, такими як: Java, Python, Scala.
- **Enterprise Architect** - це візуальне моделювання та проєктування інструменту на основі OMG UML;
- **Adobe photoshop cc 2017 –** графічний редактор, в якому можна розробити елементи дизайну для інтерфейсу системи;
- **Draw.io –** сайт для створення UML діаграм;
- **Creately.com –** ще один сайт, в якому були створені UML діаграми для даної ипломної роботи;

– **Telegram** – швидкий та надійний месенджер для комунікації із дипломним керівником.

Серед альтернативних варіантів вибору засобів розробки можна відмітити C++, C#, Python, Kotlin але вибір впас саме на мову програмування Java з наспупних причин:

- це мова, яка дає компроміс між швидкістю написання коду та швидкістю роботи програми;
- мова легко компонується з Apache Spark і Oracle DB;
- хороша ознайомленість з даною мовою програмування.

C++ - один з найкращих виборів мови програмування, коли йдеться про оптимізацію та найшвидшу роботу, але розробка на даній мові програмування займає вдвічі, втричі більше часу, що може сильно сказатися на кінцевому продукті, який не буде готовий повністю. Так як диплом буде розроблятися лише однією людиною та час досить обмежений, дану мову програмування було відкинуто з розгляду можливих мов для реалізації даної дипломної роботи.

C# - ще одна досить потужна мова програмування, що підтримується компанією Microsoft. На відміну від C++, C# дає компроміс між швидкістю написання коду та швидкістю роботи програми. Другий пункт є дуже важливим, так як алгоритм буде обробляти досить масивні дані інформації. Але вибіра на дану мову програмування не пав, тому що є досить мало хороших API для роботи з Oracle DB, так як компанія Microsoft має власну систему баз даних MS Sql. Але в даній роботі, я хотів би застосувати саме Oracle DB. Тому дана мова програмування не могла розглядатися як потенційна мова для написання дипломної роботи.

Python – одна із найпрогресивніших мов програмування, що є сьогодні на ринку. Мова дуже гнучка і легка в написанні коду, але так як всі опирації проводяться в runtime, швидкість роботи коду написаного на Python буде не

достатньо, щоб продемонструвати переваги технологій, на яких буде написана дана дипломна робота.

Kotlin [21]– новітня мова програмування, написана розрониками з компанії JetBrains. Дана мова підтримує всі бібліотеки Java, та є дуже швидкою в плані написання коду, при чому зберігає швидкість виконання коду як в Java. Причому дана мова має всі привілегії, які має вищесказана мова. Kotlin був би найкращим вибором для написання даної дипломної роботи, але так як в автора не має достатньої кількості досвіду програмування на дані мові, довелося відмовитися від цієї ідеї. Хоча, в майбутньому, я розгляну варіант, щоб переписати дипломну роботу засме на цій мові, так як вважаю, що через декілька років вона повністю замінить Java.

Як основа для зберігання інформації, було обрано зберігання в текстовому документі. Також були можливі варіанти MS SQL, MongoDB.

В основному вибір впав на зберігання в текстовому документів через постановку задачі та цілі, які були поставлені при розробці програмного забезпечення, а саме проведення дослідження.

MS SQL [22] – повна назва Microsoft Server SQL. Є досить потужною БД, яка можу зберігати Big Data. Основна мова запитів – Transact SQL, що була створена спільно з Microsoft і Sybase. Transact-SQL є реалізацією стандарта ANSI/ISO по структуруванню запитів з розширеннями. Використовується для роботи з базами даних розміром від персональних даних до крупних баз даних масштабу підприємства. Тобто, підходить для розробки даного програмного забезпечення. Але так як автор вирішив, що буде працювати з мовою Java, встановолювати базу даних MS SQL було б не найкращою ідеєю.

MongoDB [23] – це СУБД, яка не потребує описання схеми таблиць, класифікована як NoSQL. Використовує JSON-подібні документи і схему баз даних. Дана система управління баз даних була написана на мові програмування C++. Досить гарний вибір, бо ця СУБД може працювати з Hadoop, Apache Spark, але так як автор не мав жодного досвіду працювання та

встановлення даної БД, то довелося відмовитися від цієї ідеї, бо сильних переваг в плані швидкості взаємодії з програмним забезпеченням і написання коду бази даних не має.

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Так як даний продукт являє собою інструментарій роботи багаторядного методу групової організації аргументів із технологією Apache Spark то, для правильної та найшвидшої роботи даного програмного забезпечення, потрібно мати:

- персональний комп’ютер із як мінімум 2 ядрами для розпаралелення на технології Apache Spark. Чим більше ядер на комп’ютері, тим краще себе буде проявляти Apache Spark;
- як мінімум 2 ГБ оперативної пам’яті, бо кількість моделей може досягати великої кількості в буфері;
- операційну систему Windows 7/8/8.1/10, Linux, 64-bit;

Інші засоби не впливають на роботу основної програми та можуть мати будь-які версії.

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма розгортання

Діаграма розгортання (англ. deployment diagram) — це UML діаграма, що показує виконання архітектури системи, включаючи як залізо, так і середовища програмне забезпечення та взаємозв’язок між ними. Це вид структурованої діаграми, що використовується в моделюванні фізичних аспектів об’єктно-орієнтованої системи. Допомагає візуально представити елементи та комунікації між ними.

Для відображення моделювання фізичної системи була побудована структура схеми розгортання, що наведена в графічному матеріалі.

4.3.2 Діаграма класів

Діаграма класів (англ. class diagram) – одна із діаграм UML, що представляє собою діаграму зі статичною структурою, що описує систему, показуючи класи аплікації, її атрибути, операції (або методи) та відношення між об’єктами.

Діаграма класів – одна із найважливіших діаграм в об’єктно орієнтованому моделюванні. Вона використовується ддля загального концептуального моделювання структури додатку, та для детального моделювання, що буде переведено в програмний код.

Так як даний алгоритм розроблений за допомогою технології об’єктно-орієнтованого програмування, то діаграма класів є одним із найважливіших компонентів перед початком розробки коду.

Для відображення моделювання структури додатку була побудована структура класів, що наведена в графічному матеріалі.

4.3.3 Діаграма послідовності

Діаграма послідовності (англ. sequence diagram) – діаграма, що зображує взаємодію між об’єктами у послідовному порядку, наприклад порядок в якому виконується певна взяємодія відповідно до часу.

Діаграми послідовності частіше всього асоціюються з реалізаціями Use Case в логічному представленні системи, що знаходиться в розробці. Діаграми послідовності також мають назву діаграми подій, або діаграми сценаріїв.

Для відображення взаємодії між об’єктами в додатку була побудована структура послідовності, що наведена в графічному матеріалі.

4.3.4 Специфікація функцій

Таблиця 2.1

Назва класу	*Назва методу*	*Дія*
MGUA	public static void main()	Метод, з якого починається робота програми
Model	boolean isInitialModel()	Повертає true, якщо модель є початковою
Model	double getValueModel(int i)	Повертає значення і-тої моделі
Model	double getCriteria()	Повертає критерій певної моделі
Model	boolean isInitialModel()	Повертає true, якщо модель є початковою
InitialModel	double getVariable()	Повертає значення змінної
InitialModel	double getCoefficients()	Повертає значення коефіцієнтів
InitialModel	double getCriteria()	Повертає значення критерію
Criteria	double getCriteria(double[] trueY, double[] resultY)	Бере дані, які були за умовою, та отримані експериментальні дані, та на основі їх різниці отримує критерій
InitialModelOperations	void calculateModelCritrerias()	Рахує критерії моделі

Продовження таблиці 2.1

InitialModelOperations	static double[] getTrueY()	Повертає y, які були дані за умовою
ModelOperations	ModelInterface getModel()	Повертає модель: або InitialModel, або Model
BestModelsChooser	getBestModelsChooser()	Повертає найкращу модель
BestModelsChooser	void searchBestModel()	Шукає найкращу можель
BestModelsChooser	double calculateCoefficients(double a0, double a1, double a2, double a3)	Рахує коефіцієнти моделі
ModelInterface	boolean isInitialModel()	Повертає true, якщо модель є початковою
ModelInterface	double getValueModel(int i)	Повертає значення і-тої моделі
ModelInterface	double getCriteria()	Повертає значення критерію
Variable	static int getValueNumber()	Повертає номер змінної
Variable	int getVariable()	Повертає коефіцієнт, що стоїть біля змінної
Variable	double[] getVariableValues()	Повертає значення змінних в масиві

Висновок до розділу

В даному розділі було розглянуто програми, та мови програмування, за допомогою яких було реалізовано програмний продукт. Розглянуто загальні вимоги для технічних засобів.

Для кращого уявлення взаємодії об’єктів на фізичному рівні, було графічно представлено схему, на якій було вказано як взаємодіють компоненти. Було показано всі бібліотеки, операційну систему, сервер та власне, клієнта, який із всім взаємодіяв.

Перед початком програмування додатку, було розроблено структурну схему класів, в якій представлено кожен клас, методи та поля в класі та взаємодію між класами. Саме після розробки схеми класів, були виставлені пріорітети в розробці та розставлені правильні акценти, на які класи потрібно звернути більше уваги, а які можна описати помірно.

Для розуміння, як об’єкти системи взяємодіють, було розроблено структурну схему діаграми послідовностей, на якій було показано внутрішню взаємодію між класами, а саме, як користувач від запуску програми отримує готовий результат. Також, було описано кожен метод, що представлений у додатку, для розуміння за що відповідає конкретний метод та клас.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Було розроблено додаток для персонального комп’ютера, що може бути запущений на системах Windows 7 і вище, та будь-якій Unix- системі.

Для того, щоб зрозуміти як користувач взаємодіє із програмою, покажемо роботу додатку та всі можливі варіанти використання.

На рисунках 5.1 – 5.12 показаний інтерфейс програмного забузпечення та покроково розібрано процес запуску задач на багаторядний метод групового урахування аргументів.

5.1.1 Початкова сторінка

Після запуску програмного забезпечення на персональному комп’ютері, користувач потрапляє на початкову сторінку. На початковій сторінці користувач може обрати один із наступних варіантів:

- створити новий проєкт;
- оглянути покрокову інструкцію;
- подивитись відомість про автора.

Також справа доступна кнопка допомоги, при натиску на яку відкриється покрокова інструкція взаємодії із додатком.

На рисунку 5.1 зображено початкову сторінку додатку:

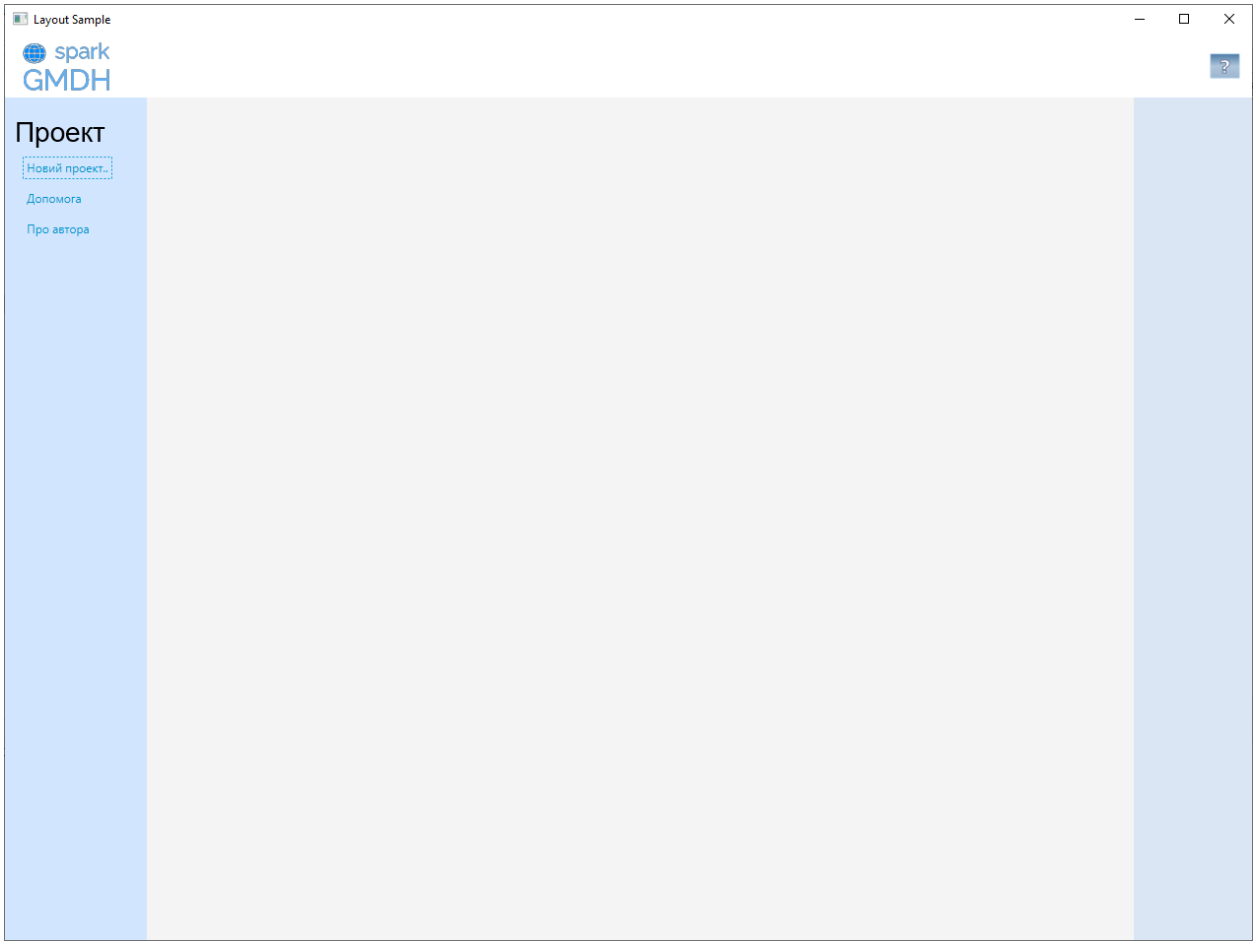


Рисунок 5.1 – Головна сторінка додатку

Наразі на головному вікні додатку ми не бачимо нічого, окрім трьох вищезгаданих пунктів, але після того, як користувач почне створювати новий проєкт, посередині екрану з'являться дані, введені користувачем, але загружені з файлу та налаштування моделі.

Перед розробкою програмного забезпечення було вирішено зробити додаток, який мав би весь потрібний функціонал на одному вікні. Таким чином нам не доведеться переносити дані між вікнами, що поліпшує процес розробки програмного забезпечення та зберігає всі дані лише в одному місці, що можу бути ще одним проявом безпеки даних.

5.1.2 Огляд покрокової інструкції

Для того, щоб більш детально ознайомитися із програмним забезпеченням, користувачеві пропонується вивчити покрокову інструкцію.

Щоб оглянути інструкцію, користувач повинен натиснути «Допомога» на головній сторінці додатку, після чого відкриється вспливаюче вікно, на якому буде детальна інформація по взаємодії із даним програмним забезпеченням.

На рисунку 5.2 зображено початкову інструкцію програмного забезпечення:

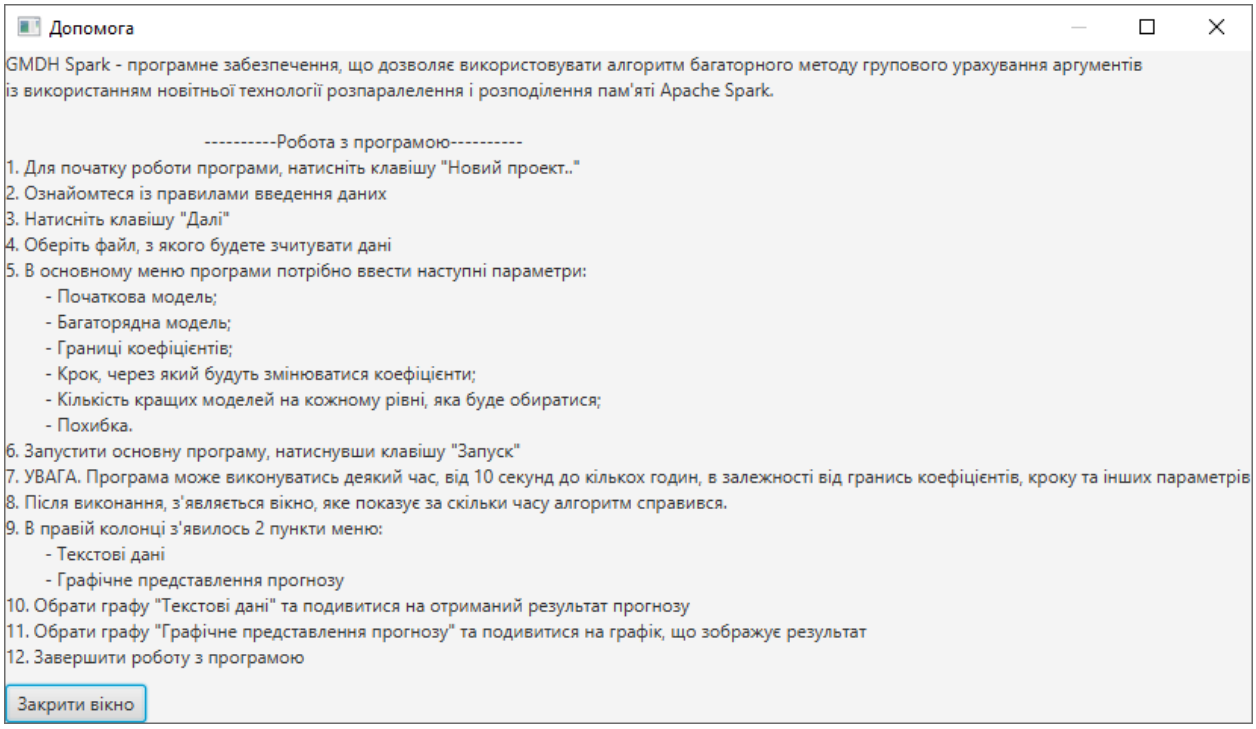


Рисунок 5.2 – Початкова інструкція програмного забезпечення

Як зображено на рисунку, покрокова інструкція фактично огортає користування зі всією програмою. Було вирішено розробити саме такий вид допомоги для користувача, а не розбиття по пунктах із посиланнями, так як в користувача може виникнути одразу кілька питань щодо того, як взаємодіяти із програмним забезпеченням.

Таким чином, після прочитання даного вікна із допомогою, користувач може із легкістю починати роботу з програмою

5.1.3 Відомість про автора

Щоб оглянути інформацію та контакти людини, яка створила програмне забезпечення, користувачеві потрібно натиснути «Про автора», після чого

з’явиться спливаюче вікно із повною інформацією про автора програмоного забезпечення.

На рисунку 5.3 зображено інформацію про автора програмного забезпечення:

Рисунок 5.3 – Інформація про автора

Із розробником даного дипломного проекту можна зв’язатись, використовуючи електронну пошту, або сучасні месенджери. Також представлений github розробника, на якому можна знайти й інші проєкти.

5.1.4 Створення нового проєкту

При натисканні на «Новий проєкт..», з’являється спливаюче вікно, яке пропонує 2 опції:

- зчитати дані досліджень із жорсткого диску;
- ввести дані вручну.

Перед тим, як обрати один із пунктів, обов’язково потрібно ознайомитися із правилами введення даних.

На рисунку 5.4 зображено вікно із вибором зчитування:

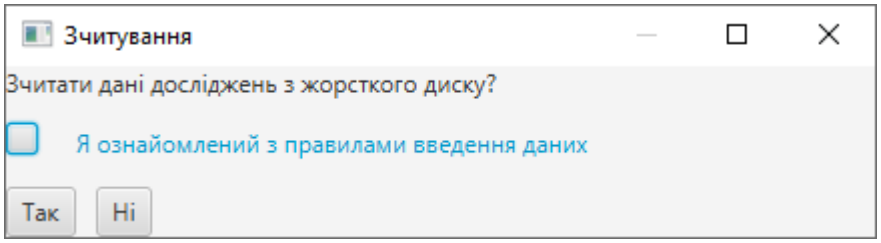


Рисунок 5.4 – Вікно із вибором зчитування

Так як дані на вхід зчитуються в програмі за певним принципом, додаток не дозволить продовжити роботу без обов'язкового ознайомлення із правилами введення даних. Тому що користувач, не прочитавши правила, не зможе коректно взаємодіяти із програмним забезпеченням.

5.1.4.1 Правила введення даних

У вікні із вибором зчитування, користувач може натиснути на посилання із більш детальним ознайомленням.

На рисунку 5.5 зображено вікно із правилами введення даних:

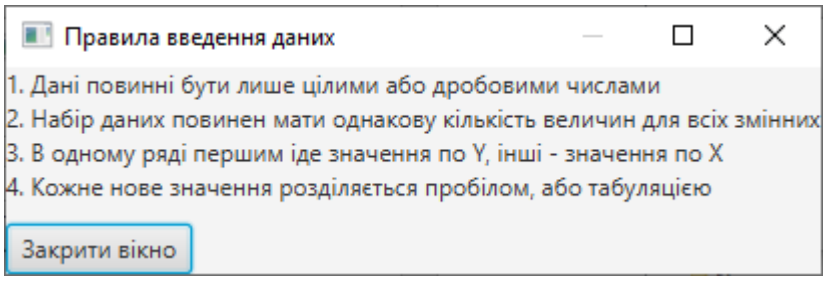


Рисунок 5.5 – Вікно із правилами введення даних

В цьому вікні користувач має змогу ознайомитися як саме він має взаємодіяти із програмою і як саме мають виглядати дані на вхід.

Наприклад, якщо користувач розділить дані за допомогою коми чи іншого символа, програма буде не в змозі зчитати такі дані і в результаті видасть помилку.

Також для користувача важливо пам'ятати, що перший стовпець – це значення по Y, тобто значення, які ми будемо шукати за допомогою алгоритму, тоді як всі інші стовпці відповідають виключно за аргументи, якими можуть бути різні дані.

5.1.4.2 Додавання даних досліджень

Після того, як користувач обрав яким чином додавати дані дослідження, на головному вікні з’являються самі дані і різні параметри, які потрібно ввести, для того, щоб запустити алгоритм в дію. Потрібно також відмітити, що від введених користувачем параметрів залежить точність прогнозу.

Користувач може ввести наступні параметри:

- початок діапазону коефіцієнтів;
- кінець діапазону коефіцієнтів;
- крок зміни коефіцієнтів;
- кількість нових моделей, що мають генеруватися на наступних кроках;
- точність критерію;

На рисунку 5.6 зображено головне вікно із параметрами та зчитаними даними:

Рисунок 5.6 – Головне вікно із параметрами та зчитаними даними

Як показано на рисунку, користувачеві буде представлено обрати початкову та багаторядну модель. Слід замітити, що вибір моделі суттєво впливає на якісний результат прогнозу. Так, наприклад, для вибору початкової моделі було представлено наступні моделі:

- a + b*x; (5.1)
- 10a + 0.1b*x; (5.2)
- 0.1a + 10b*x; (5.3)
- a + b*cos(x); (5.4)
- a + b*sin(x); (5.5)

Користувач може обрати одну із наступних багаторядних моделей:

- $a + b*x_1 + c*x_2 + d*x_1*x_2$; (5.6)
- $100a + 10b*x_1 + 10c*x_2 + d*x_1*x_2$; (5.7)
- $15a + 10b*x_1 + 5*c*x_2 + 0.01*d*x_1*x_2$; (5.8)

- $1000a + 200b*x_1 + 200c*x_2 + 0.2d*x_1*x_2$; (5.9)
- $a + b*x_1 + c*x_2 + d*\cos(x_1*x_2)$; (5.10)
- $a + b*x_1 + c*x_2 + d*\sin(x_1*x_2)$; (5.11)

Так як можна обрати як початкову, так і багаторядну модель, то в поєднанні ми отримуємо велику кількість можливих функції. Таким чином можна спробувати різні комбінації і отримати найбільш оптимальну для даного набору.

Також слід звернути увагу, що навіть незважаючи на пришвидшення за допомогою технології Spark, алгоритм залишається досить повільним, тобто важливо обрати не досить широкий діапазон, так як вираховуватися алгоритм із широким діапазоном може і декілька років.

Вибір кількості кращих моделей суттєво впливає на точність алгоритму, так як чим більше моделей піде на наступний рівень, тим більший шанс знайти саме оптимальну модель. В даному застосуванні можна обрати до 5-ти моделей, так як при збільшенні пошуку хоча б на 1 модель час виконання алгоритму зростає експоненційно.

5.1.4.3 Запуск моделі

Після заповнення всіх параметів коректно, користувач запускає модель, натискаючи клавішу «Run».

Алгоритм буде працювати деякий час, цей час залежить від обраних параметрів. Користувачеві потрібно дочекатися закінчення роботи алгоритму, після чого він отримає відповідне вікно.

На рисунку 5.7 зображено спливаюче вікно, що з’являється на головному екрані із часом роботи алгоритму.

Рисунок 5.7 – Вспливаюче вікно із часом роботи алгоритму

Поява даного вікна означає, що алгоритм завершив свою роботу і в програмі уже містяться результати.

Після ознайомлення із часом, користувач мусить закрити це вікно для того, щоб почати оглядати результати роботи алгоритму.

5.1.4.4 Результати роботи алгоритму в текстовому вигляді

На панелі справа з’явилося вікно для представлення даних у текстовому вигляді «Data». При натиску на дану кнопку, користувач отримує шукану модель, а також моделі для кожного із аргументів, які теж були побудовані алгоритмом.

На рисунку 5.8 зображено текстове представлення даних, обраховане алгоритмом:

Рисунок 5.8 – Вікно із даними роботи алгоритму в текстовому вигляді

Також слід відмітити, що користувачеві надається інформація про критерії під кожним із аргументів. Таким чином, користувач може оцінити наскільки точно алгоритм розрахував прогноз.

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

За мету випробовувань ставиться перевірка програмного продукту на правильну роботу, перевірка коректності зчитування данних програмним продуктом, оцінка правильності отриманих даних за допомогою різних алгоритмів.

5.2.2 Загальні положення

Випробовування здійснюються згідно з наступними документами:

- ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
- ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

За результатами тестів, була повністю перевірена правильна функціональність моделі. У таблицях нижче наведено перелік всіх випробовувань. Було також наведено опис тестів.

Таблиця 5.1 – Ознайомлення з додатком

Дія:	Перегляд допомоги користувача	
	Очікуваний результат	Результат тесту
Передумова		
Відкрити додаток на персональному комп’ютері	Додаток відкритий, користувач знаходиться на головній сторінці	пройдений
Кроки тесту		
Натиснути на клавішу «Допомога»	Відкривається вікно із детальною інформацією про користування програмним забезпеченням	пройдений
Післяумова		
Натиснути на клавішу «Закрити вікно»	Вікно було закрито, користувач повернувся на головне вікно додатку	пройдений

Таблиця 5.2 – Ознайомлення з інформацією про автора

Дія:	Перегляд інформації про автора	
	Очікуваний результат	Результат тесту
Передумова		
Відкрити додаток на персональному комп’ютері	Додаток відкритий, користувач знаходиться на головній сторінці	пройдений
Кроки тесту		
Натиснути на клавішу «Інформація про автора»	Відкривається вікно із детальною інформацією про автора із посиланням на github	пройдений
Післяумова		
Натиснути на клавішу «Закрити вікно»	Вікно було закрито, користувач повернувся на головне вікно додатку	пройдений

Таблиця 5.3 – Ознайомлення з інформацією про автора

Дія:	Додавання даних до прогнозу	
	Очікуваний результат	Результат тесту
Передумова		
На головному вікні додатку натиснути клавішу «Новий проєкт»	Відкрите вікно, яке пропонує ввести дані вручну, або зчитати дані з файлу	пройдений
Кроки тесту		

Продовження таблиці 5.3

Натиснути на клавішу, що відповідає за зчитування даних з файлу	Відкривається вікно, в якому користувачеві потрібно обрати файл формату *.csv	пройдений
Післяумова		
Обрати файл та натиснути клавішу «Відкрити»	Дані, що були представлені користу-вачем з’являються на головному вікні. З’явля-ються параметри, які потрбіно заповнити, щоб запустити модель	пройдений

Таблиця 5.4 – Ознайомлення з інформацією про автора

Дія:	Додавання даних до прогнозу	
	Очікуваний результат	Результат тесту
Передумова		
Відкрити дані з жорсткого диску, або ввести дані вручну	Дані з’являються на головному вікні додатку. З’язлаються поля, в які потбірно ввести параметри пошуку моделей	пройдений
Кроки тесту		
Заповнити всі представлені параметри та натиснути клавішу «Run»	Алгоритм починає вираховувати найкращу модель. Після обрахунку спливає вікно, на якому зазначено скільки часу алгоритм рахував.	пройдений

Продовження таблиці 5.4

Післяумова		
Закрити вікно, що показує час прорахунку алгоритму.	Справа на головному вінкі додатку з'являються опції як користувач може переглянути отримані дані прогнозу	пройдений

Таблиця 5.5 – Ознайомлення з інформацією про автора

Дія:	Додавання даних до прогнозу	
	Очікуваний результат	Результат тесту
Передумова		
Закрити вікно, що показує час прорахунку алгоритму	Справа на головному вінкі додатку з'являються опції як користувач може переглянути отримані дані прогнозу	пройдений
Кроки тесту		
Справа натиснути на клавішу «Data»	З'являється вікно, в якому показана головна модель, що була прорахована алгоритмом, критерій моделі та також представлені моделі для алргументів	пройдений
Післяумова		
Закрити вікно із даними	Користувач повертається на головне вікно додатку	пройдений

Таблиця 5.6 – Ознайомлення з інформацією про автора

Дія:	Додавання даних до прогнозу	
	Очікуваний результат	Результат тесту
Передумова		
Закрити вікно, що показує час прорахунку алгоритму	Справа на головному вінкі додатку з’являються опції як користувач може переглянути отримані дані прогнозу	пройдений
Кроки тесту		
Справа натиснути на клавішу «Line»	З’являється вікно, в якому представлений графік із поточними даними та із даними прогнозу	пройдений
Післяумова		
Закрити вікно з графіком	Користувач повертається на головне вікно додатку	пройдений

Висновок до розділу

В даному розділі було повністю описано користувацьку складову дипломного проєкту. Було показано на скріншотах всі можливі випадки використання програмного забезпечення, протестовано функції додатку.

Також було пояснено як користуватися даним програмним забезпеченням та показано які можливі функції можна обрати для кожної моделі, як для початкової, так і для багаторядної.

ЗАГАЛЬНІ ВИСНОВКИ

Даний дипломний проєкт розглядає роботу багаторядного методу групового урахування аргументів та генетичного алгоритму, реалізованою за допомоги технології Apache Spark.

Дипломний проєкт був присвячений саме даним алгоритмам, тому що за мету було поставлено пришвидшити роботу багаторядного методу групового урахування аргументів, так як результати правильно реалізований алгоритм із правильно підібраними функціями дає дуже добрий, але для отримання хороших результатів, потрібно взяти великий діапазон коефіцієнтів, що значно сповільняє знаходження результатів. Але, за допомогою додатку від компанії Apache, було здійснено спробу збільшити швидкість алгоритму, навіть на великих діапазонах.

Також було реалізовано генетичний алгоритм, так як за допомогою цього алгоритму також можна спрогнозувати рішення для бізнесу, медицини, оборони і так далі. Генетичний алгоритм з’явився на 20 років пізніше за алгоритм Івахненка МГУА, але по суті генетичний алгоритм продовжує ідею методу групового урахування аргументів.

В розділі програмного та технічоного забезпечення було детально розібрано архітектуру як програмного продукту, так і технологій, що використовувалися при розробці.

Таким чином, було створено додаток, в якому користувач може самостійно протестувати свої дані і зробити висновки щодо швидкості роботи алгоритмів, реалізованих на даній технології.

ПЕРЕЛІК ПОСИЛАНЬ

1. Big data simulation [Електронний ресурс] – Режим доступу: https://www.ibm.com/analytics/hadoop/big-data-analytics
2. Модель Хольга-Уінтерса: Математичні аспекти і комп'ютерна реалізація [Електронний ресурс] – Режим доступу: https://journal.tltsu.ru/rus/index.php/VNSEM/article/view/7868.
3. Статистичний аналіз часових рядів авторегресії і ковзункового середнього / А.Ф. Тараскін. – Самарський державний аерокосмічний університет. – Самара : СДАУ, 1998. – 64 с.
4. Прийняття рішень на основі самоорганізації / А.Г. Івахненко, Ю.П. Зайченко, В.Д. Дімітров. – Москва: Видавництво «Радянське радіо», 1976. – 276 с.
5. Learning Spark: Lightning-Fast Big Data Analysis [Електронний ресурс] – Режим доступу: http://index-of.co.uk/Big-Data-Technologies/Learning.
6. What is YARN [Електронний ресурс] – Режим доступу: https://www.encyclopedia.com/sports-and-everyday-life/fashion-and-clothing/textiles-and-weaving/yarn.
7. What is Apache MapReduce [Електронний ресурс] – Режим доступу: https://www.ibm.com/analytics/hadoop/mapreduce.
8. Apache Pig Tutorial [Електронний ресурс] – Режим доступу: https://www.thoughtworks.com/radar/tools/apache-pig.
9. Apache Hive – What It Is, What It Does, and Why It Matters? [Електронний ресурс] – Режим доступу: https://mapr.com/products/apache-hive/.
10. Apache Sqoop [Електронний ресурс] – Режим доступу: https://sqoop.apache.org/.
11. GoodData [Електронний ресурс] – Режим доступу: https://www.gooddata.com/.

12. Python [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Python.
13. Ruby [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Ruby.
14. R [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/R_(programming_language).
15. Scala [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Scala.
16. GMDH Streamline [Електронний ресурс] – Режим доступу: https://gmdhsoftware.com/.
17. MPI [Електронний ресурс] – Режим доступу: https://pl.wikipedia.org/wiki/MPI.
18. iPyParallel [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/IPython.
19. Олейник, Ю. А. "Применение метода группового учета аргументов с оператором сдвига для прогнозирования финансовых показателей местных бюджетов/Олейник ЮА, Томашевский ВН, Виноградов АН." Проблемы управления и информатики. К.: Институт кибернетики им. ВМ Глушкова НАН Украины 3 (2008): 143-151.
20. Oliynyk, Yu O., O. M. Vinogradov, and K. M. Krasovsky. "DISTRIBUTED COMPUTATION SYSTEM OF FORECASTING INDICES BY ADAPTIVE GMDH." Naukovi visti NTUU-KPI 2007.5 (2007).
21. Kotlin [Електронний ресурс] – Режим доступу: https://pl.wikipedia.org/wiki/Kotlin.
22. MS SQL [Електронний ресурс] – Режим доступу: https://pl.wikipedia.org/wiki/MSSQL.
23. MongoDB [Електронний ресурс] – Режим доступу: https://pl.wikipedia.org/wiki/MongoDB.

Model.class

```
public abstract class Model implements Comparable<Model> {

   double innerCriteria;
   private double regulationCriteria=-1;
   public abstract double getValueModel(int i, Map<X,double[]> checkingXs);
   public abstract double[] getCoefficients();
   public abstract Function getFunction();
   public double getInnerCriteria(){
      return innerCriteria;
   }
   public double getRegulationCriteria(){
      if(regulationCriteria==-1)
         throw new RuntimeException("RegularCtiteria is not set.");
      return regulationCriteria;
   }
   public void setRegulationCriteria(double[] checkingYValues,Map<X,double[]> checkingXs)
{
      regulationCriteria = Criteria.getRegulationCriteria(checkingYValues,this,checkingXs);
   }


   @Override
   public int compareTo(Model o) {
      return Double.compare(this.getRegulationCriteria(),o.getRegulationCriteria());
   }

}
```

InitialModel.class

```
package model;

import data.DataType;
import functions.initial_functions.CommonInitialFunction;
import functions.initial_functions.CommonInitialFunction2;
import functions.initial_functions.InitialFunction;
import variable.X;

import java.util.Map;

// Initial model is the model we choose when start building big polynomiums
public class InitialModel extends Model {

   // Every initial model has its own variable: For example x7
   private X x;
   // Every initial model has 2 coeficients: a0 and a1
   private double[] coefficients;

   // Initial Common function - TODO you can change it to some other
   InitialFunction function = new CommonInitialFunction2();
```

```
    // To set up initial model you need a variable, coeficients and the criteria of the model
    public InitialModel(X x, double[] coefficients, double criteria){
        this.x= x;
        this.coefficients = coefficients;
        this.innerCriteria = criteria;
    }


    public X getX() {
        return x;
    }

    @Override
    public double[] getCoefficients() {
        return coefficients;
    }


    @Override
    public boolean equals(Object obj) {
        return x == ((InitialModel)obj).getX() && innerCriteria == ((InitialModel)obj).get-
InnerCriteria();
    }



    @Override
    public String toString() {
        return function.functionToString(x,coefficients);

    // the method to get value from particular experiment(e.g. 1+3x2)(the eight value of x2)
    @Override
    public InitialFunction getFunction(){
        return function;
    }

    @Override
    public double getValueModel(int i, Map<X,double[]> dataXs){
        // getX() - we got x3 for example, getXValue(34) - we got the value of x3 in 34 case(row)
        return function.getResult(getX().getXValue(i, dataXs),coefficients);

        //return coefficients[0]+coefficients[1]*getX().getXValue(i);
    }

}
```

ComplexModel.class

```
public class ComplexModel extends Model {

    // We will set it later, because we don't know yet what coefficients are the best
```

```
    private Model modelS;
    private Model modelT;
    private TupleCoefficients coefficients;

    private ComplexFunction function = new CommonComplexFunction2();

    public ComplexModel(Model modelS, Model modelT, TupleCoefficients coefficients, double
criteria){
        this.modelS = modelS;
        this.modelT = modelT;
        this.coefficients = coefficients;
        this.innerCriteria = criteria;
    }

    @Override
    public ComplexFunction getFunction(){
        return function;
    }

    @Override
    public double getValueModel(int i, Map<X,double[]> checkingXs){
        return function.getResult(modelS.getValueModel(i,checkingXs),modelT.getValue-
Model(i,checkingXs),coefficients.getCoefficients());
    }


    @Override
    public double[] getCoefficients() {
        return coefficients.getCoefficients();
    }

    @Override
    public String toString() {
        return function.functionToString(modelS,modelT,coefficients.getCoefficients());
    }

}


ModelOperations.class
abstract class ModelOperations {

    abstract double calculateCoefficients(double[] coefs, Map<X,double[]> dataXs, X... x);
    private int coefVariable = 0;
    abstract double calculateMinCriteria(double[] coefficients, double minimumCriteria,
TupleCoefficients minCoefficients, int loopNumber,Map<X,double[]> dataXs, X... x);
}
```

InitialModelOperations.class

```
public class InitialModelOperations extends ModelOperations {

    // There will be restricted class of initial models. In the case I consider first:
    // 10 initial models that look:
    // 1: a0+a1*x1
    // 2: a0+a1*x2
    // 3....
    // N: a0+a1*xN
    private ArrayList<Model> models = new ArrayList<>();
    InitialFunction function = MyData.MY_INITIAL_FUNCTION;
    private double[] yValues = null;
    private Map<X, double[]> xValues = new LinkedHashMap<>();

    // How many y-s we have. How many experiments we managed to make
    //private int experimentQuantity = 0;

    // model.ComplexModel will look:
    // y = a0 + a1*x1 and so on
    // starting coefficients
    private double[] coefficients = new double[MyData.INITIAL_COEF_NUM];


    public InitialModelOperations(double[] yValues, Map<X, double[]> xValues) {
        this.yValues = yValues;
        this.xValues = xValues;
    }

    // In this method we will pick the lowest criteria for each model:
    public ArrayList<Model> calculateModelCritrerias() {



        for (X x : xValues.keySet()) {
            TupleModel<Double, TupleCoefficients> bestCoefForModel = makeCoefficients(x);
            models.add(new InitialModel(x, bestCoefForModel.getCoefficients().getCoefficients(),
bestCoefForModel.getCriteria()));

        }

        return models;
    }


    private TupleModel<Double, TupleCoefficients> makeCoefficients(X x) {
        for (int i = 0; i < MyData.INITIAL_COEF_NUM; i++) {
            coefficients[i] = MyData.round(MyData.START.doubleValue()); //-(MyData.N * My-
Data.STEP) / 2;
        }
        // TODO: Here I consider only common situation when Initial coefficient number = 2
        // TODO: Maybe you'll wnt to do it more complex(unnesassary)
```

```
        if (MyData.INITIAL_COEF_NUM != 2)
            throw new NotCorrectInput();

        double minimumCriteria = Double.MAX_VALUE;
        TupleCoefficients minimumCoefficients = new TupleCoefficients();
        minimumCriteria = calculateMinCriteria(coefficients, minimumCriteria,minimumCoeffi-
cients, coefficients.length, xValues, x);

        return new TupleModel<>(minimumCriteria, minimumCoefficients);
    }

    @Override
    double calculateCoefficients(double[] coefs, Map<X,double[]> dataXs, X... xArray) {
        if (coefs.length != MyData.INITIAL_COEF_NUM)
            throw new NotCorrectInput();

        X x = xArray[0];

        int numberOfValues = yValues.length;
        double resultY[] = new double[numberOfValues];

        int counter = 0;
        // i - here is number or expreriment(номер дослідження)

        for (int i = 0; i < numberOfValues; i++) {
            resultY[i] = function.getResult(xValues.get(x)[i], coefs);
        }
        return Criteria.getInnerCriteria(yValues, resultY);

    }

    @Override
    double calculateMinCriteria(double[] coefficients, double minimumCriteria, TupleCoeffi-
cients minCoefficients, int loopNumber,Map<X,double[]> dataXs, X... x) {
        boolean entry = false;
        for (int i = 0; i < MyData.N; i++) {
            for (int j = 0; j < MyData.N; j++) {
                double currentCriteria;
                coefficients[0] += MyData.STEP.doubleValue();

                currentCriteria = calculateCoefficients(coefficients,dataXs,x);

                // if the current criteria with such coeffisients is less then the minumim we have, then
                if (currentCriteria < minimumCriteria) {
                    minimumCriteria = currentCriteria;
                    minCoefficients.setCoefficients(coefficients);
                    entry = true;
                }
            }
            coefficients[1] += MyData.round(MyData.STEP.doubleValue());
```

```
            coefficients[0] = MyData.round(MyData.START.doubleValue());//-(MyData.N * My-
Data.STEP) / 2;
        }
        if(!entry)
            minCoefficients.setCoefficients(coefficients);
        return minimumCriteria;
    }

}
```

Data.class

```
public class MyData {

    // TODO Here will be set N and STEP, but in future you need to transport it to the better
place
    public static InitialFunction MY_INITIAL_FUNCTION = new CommonInitialFunction2();
    public static ComplexFunction MY_COMPLEX_FUNCTION = new CommonComplexFunc-
tion2();
    public static BigDecimal START = new BigDecimal(-0.8);
    public static BigDecimal FINISH = new BigDecimal(0.8);
    // Interval between steps we do
    public static BigDecimal STEP =new BigDecimal(0.1);
    public static int N = (FINISH.subtract(START)).divide(STEP,2, RoundingMode.CEIL-
ING).intValue();

    // Number of best models that will be chosen on every step
    public static int NUMBER = 4;
    // Number of different coefficients in InitialModel
    public static final int INITIAL_COEF_NUM = 2;
    // Number of different coefficients in ComplexModel
    public static final int COMPLEX_COEF_NUM = 4;
    // Criterion
    public static double CRITERION = 0.001;
    // For how many dots are we Forecasting
    public static int FORECAST_NUM = 5;
    // Proportion Training to Checking Data:
    private static final double PROPORTION = 0.5;
    // Rounding a number in Java Program
    private static final int PLACES = 3;


    // The map in which we will store our data to analyse
    // data.YValue = our y value
    // double[] - set of x values
    private static Map<YValue,double[]> dataMap = new LinkedHashMap<>();
    private static Map<X,double[]> xTrainingMap = new LinkedHashMap<>();
    private static Map<X,double[]> xCheckingMap = new LinkedHashMap<>();
    private static int trainingValuesNumber;
    private static int checkingValuesNumber;

    // These are values of Y. We will extract them to the double array
```

```
// when the data will be exported
private static double[] yTrainingValues = null;
private static double[] yCheckingValues = null;
// Whether data is exported or not
private static boolean dataIsExported = false;

public static void exportDataText(String text){
    Scanner dataScanner = new Scanner(text);

    // Special pattern go obtain only numbers without spaces
    Pattern dataObtainer = Pattern.compile("\\s+");

    // Starting to write our data into Map
    while (dataScanner.hasNext()){
        // Row which contains y and all the x-s
        String dataRow = dataScanner.nextLine();
        String[] outputData = dataObtainer.split(dataRow);
        // First item of output row will be y(yValue), all the next - x
        YValue yValue = new YValue(Double.parseDouble(outputData[0]));
        double[] xArray = new double[outputData.length-1];
        for (int i = 1; i < outputData.length; i++) {
            xArray[i-1] = Double.parseDouble(outputData[i]);
        }
        dataMap.put(yValue,xArray);
    }
    dataIsExported = true;

    trainingValuesNumber = (int)(dataMap.size()*PROPORTION);
    checkingValuesNumber = dataMap.size()-trainingValuesNumber;

    extractYVariables();
    extractXVariables();
}
public static void exportData(String filename) throws FileNotFoundException {

    Scanner dataScanner = new Scanner(new FileReader(filename));

    // Special pattern go obtain only numbers without spaces
    Pattern dataObtainer = Pattern.compile("\\s+");

    // Starting to write our data into Map
    while (dataScanner.hasNext()){
        // Row which contains y and all the x-s
        String dataRow = dataScanner.nextLine();
        String[] outputData = dataObtainer.split(dataRow);
        // First item of output row will be y(yValue), all the next - x
        YValue yValue = new YValue(Double.parseDouble(outputData[0]));
        double[] xArray = new double[outputData.length-1];
        for (int i = 1; i < outputData.length; i++) {
            xArray[i-1] = Double.parseDouble(outputData[i]);
        }
```

```
            dataMap.put(yValue,xArray);
        }
        dataIsExported = true;

        trainingValuesNumber = (int)(dataMap.size()*PROPORTION);
        checkingValuesNumber = dataMap.size()-trainingValuesNumber;

        extractYVariables();
        extractXVariables();
    }
    private static void extractYVariables(){
        if(!dataIsExported)
            throw new DataIsNotExported();

        yTrainingValues = new double[trainingValuesNumber];
        yCheckingValues = new double[checkingValuesNumber];
        int yCounter = 0;
        Iterator<YValue> yIterator = dataMap.keySet().iterator();

        while (yCounter<trainingValuesNumber && yIterator.hasNext()){
            yTrainingValues[yCounter++] = yIterator.next().value;
        }

        yCounter = 0;
        while (yCounter<checkingValuesNumber && yIterator.hasNext()){
            yCheckingValues[yCounter++] = yIterator.next().value;
        }
    }
    private static void extractXVariables(){
        if(!dataIsExported)
            throw new DataIsNotExported();

        Iterator<Map.Entry<YValue,double[]>> iterator = null;

        // For training data
        for (int i = 0; i < getXnumber(); i++) {
            double[] Xi = new double[trainingValuesNumber];
            int counter = 0;

            iterator = dataMap.entrySet().iterator();
           while (counter<trainingValuesNumber && iterator.hasNext()){
              Xi[counter++] = iterator.next().getValue()[i];
           }
//
//          for (Map.Entry<YValue,double[]> entry : dataMap.entrySet()){
//              Xi[counter++] = entry.getValue()[i];
//          }
            xTrainingMap.put(X.getX(i),Xi);
        }

        Iterator<Map.Entry<YValue,double[]>> checkingIterator=null;
```

```
        // For checking data
        for (int i = 0; i < getXnumber(); i++) {
            double[] Xi = new double[checkingValuesNumber];
            int counter = 0;

            checkingIterator = dataMap.entrySet().iterator();
            int startingNumber = 0;
            while (startingNumber++<trainingValuesNumber && checkingIterator.hasNext())
                checkingIterator.next();

            while (counter<checkingValuesNumber && checkingIterator.hasNext()){
                Xi[counter++] = checkingIterator.next().getValue()[i];
            }
//
//          for (Map.Entry<YValue,double[]> entry : dataMap.entrySet()){
//              Xi[counter++] = entry.getValue()[i];
//          }
            xCheckingMap.put(X.getX(i),Xi);
        }
    }
    // Renurns Y-s from data as array
    public static double[] getTrainingY(){
        if(!dataIsExported)
            throw new DataIsNotExported();

        return yTrainingValues;
    }

    public static Map<X, double[]> getTrainingXs() {
        return xTrainingMap;
    }

    public static double[] getCheckingY(){
        if(!dataIsExported)
            throw new DataIsNotExported();

        return yCheckingValues;
    }

    public static Map<X, double[]> getCheckingXs() {
        return xCheckingMap;
    }

    public static double round(double value) {
        int places = PLACES;
        if (places < 0) throw new IllegalArgumentException();

        BigDecimal bd = new BigDecimal(value);
        bd = bd.setScale(places, RoundingMode.HALF_UP);
        return bd.doubleValue();
    }
```

```
    // getX(3) - get array of all x3
    public static double[] getTrainingX(int i){
        return xTrainingMap.get(X.getX(i));
    }
    public static double[] getCheckingX(int i){
        return xCheckingMap.get(X.getX(i));
    }

    // Number of different X-s(x1,x2,x3...x10)
    public static int getXnumber(){
        return dataMap.values().iterator().next().length;
    }
}
```

CommonInitialFunction.class

```
public class CommonInitialFunction extends InitialFunction {
    @Override
    public double getResult(double x, double... coefs) {
        return MyData.round(coefs[0])+MyData.round(coefs[1])*x;
    }

    @Override
    public String functionToString(X x, double... coefs) {
        for (int i = 0; i < coefs.length; i++) {
            coefs[i] = (double) Math.round(coefs[i] * 1000) / 1000;
        }
        StringBuilder funcBuilder = new StringBuilder();
        funcBuilder.append("(");
        funcBuilder.append(MyData.round(coefs[0]));
        if(coefs[1]>=0)
            funcBuilder.append("+");
        funcBuilder.append(MyData.round(coefs[1]));
        funcBuilder.append("*");
        funcBuilder.append(x);
        funcBuilder.append(")");

        return funcBuilder.toString();
    }
}
```

Додаток А

Тексти програмного коду

Комплекс задач прогнозування часових рядів з використанням технології Spark

(Найменування програми (документа))

DVD-R

(Вид носія даних)

10 арк, 55 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проекту	**В.о. завідувача кафедри**
_____ *Ю.О. Олійник*	_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)	(підпис) (ініціали, прізвище)
“16” квітня 2019 р.	“17” квітня 2019 р.

Комплекс задач прогнозування часових рядів із виокристанням технолгії Apache Spark

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП ІС-5220.1181-с.ТЗ*

на 10 сторінках

Київ – 2019 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: Комплекс задач прогнозування часових рядів з використанням технології Apache Spark.

Умовне позначення: GMDH Spark.

1.2 Найменування організації-замовника та організацій-учасників робіт

Замовником проекту є кафедра Автоматизованих систем обробки інформації та управління НТУУ «КПІ ім. Ігоря Сікорського». Представником замовника є Олійник Юрій Олександрович.

Розробником системи є студент факультету інформатики та обчислювальної техніки НТУУ «КПІ ім. Ігоря Сікорського»: студент групи ІС-52 Олійник Ярослав Миколайович.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створенні проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над створенням системи – 5 лютого 2019 року.

Плановий термін по закінченню роботи над створенням системи – не пізніше 1 червня 2019 року.

2 ПРИЗНАЧЕННЯ І МЕТА СТВОРЕННЯ КОМПЛЕКСУ ЗАДАЧ

2.1 Призначення системи

Призначенням системи є створення комплексу задач з прогнозуваня часових рядів за допомогою технології Apache Spark.

2.2 Цілі створення системи

Цілями розробки є:

- зменшити вартість програм для прогнозування на ринку;
- поліпшити час роботи даного алгоритму, використовуючи новітню технологію Apache Spark;
- спростити дослідження часових рядів для науковців, зробивши зручний та зрозумілий інтерфейс користувача.

Для досягнення поставлених цілей необхідно розв’язати наступні задачі:

- побудувати план реалізації алгоритму групового урахування аргументів;
- реалізувати багаторядний алгоритм групового урахування аргументів;
- зробити написаний код максимально гнучким до змін різних параметрів та початкових рівнянь;
- побудувати модель системи взаємодії користувача із програмою простою і вобночас функціональною;
- реалізувати модель системи взаємодії користувача із програмою для персонального комп’тера;
- провести повне тестування системи;

- написати вичерпну інструкцію користувача для правильного і ефективного користування програмним забезпеченням.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Щоб використовувати систему, користувач може мати одну із наступних встановлених операційних систем:

- Windows 7;
- Windows 8;
- Windows 8.1;
- Windows 10;
- будь-яка операційна система із сімейства Unix для персональних комп'ютерів;

Працювати з системою можна користувачам, що встановили даний продукт. Після того, як користувач створив систему, він має доступ до функціоналу системи, має змогу підгружати файл із даними для дослідження, або вводити дані вручну, налаштовувати параметри системи, зокрема:

- початкові значення пошуку коефіцієнтів;
- кінцеві значення пошуку коефіцієнтів;
- крок зміни коефіцієнтів;
- обрання початкової моделі;
- обрання багаторядної моделі;
- присвоєння похибки, яка можу бути допущена;
- обрання кількості найкращих моделей, що будуть обиратися на кожному кроці.

Об'єктом автоматизації є комплекс задач з прогнозування часових рядів.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Система має виконувати наступні функції:

1) Створення нових прогнозів;
2) отримання даних прогнозування;
3) отримання параметрів, зазначених користувачем, а саме:
   - початкові та кінцеві значення прогнозу;
   - крок зміни коефіцієнтів;
   - обрання моделей;
   - обрання кількості найкращих моделей;
   - присвоєння похибки.
4) переробка даних програмою та присвоєння всіх введених даних до відповідних у змінних у програмі;
5) обрахування алгоритму:
   а) створення початкових моделей;
   б) обрахування критерію початкових моделей;
   в) створення багаторядних моделей;
   г) обрахування критерію багаторядних моделей;
   д) порівняння кретирію найкращих моделей на попередньому кроці і на поточному кроці.

   Алгоритм буде рахувати наступний рівень моделей до тих пір, поки критерій не погіршиться;
6) вивід результатів користувачеві;
7) показ графіку користувачеві.

Вимоги до надійності

Програма повинна відповідати всім фукнціональни вимогам, вказаним вище.

Програма повинна відповідати користувачеві на некоректно введені дані, видаючи на головний екран відповідне повідомлення.

При збоях в роботі апаратних засобів, програма повинна коректно завершити роботу, зберігши проміжні результати.

Продукт водночас повинен бути і надійним і функціональним. У разі виникнення збоїв в програмі, потрібно сповіщати користувача, вказавши причину хбою та шляхи вирішення проблеми. Будь-які непередбачені ситуації мають бути задокументовані у звіті, який при необхідності надсилається розробнику для визначення причини збою в роботі та усуненні помилок, які могли привести до нестабільної роботи програмного продукту.

4.2 Вимоги до складу і параметрів технічних засобів

Для правильної роботи системи до складу технічних засобів повинен входити комп’ютер, що має конфігурацію наведену нижче:

- комп’ютер з такою конфігурацією:
    а) процесор з тактовою частотою не нижче 2 ГГц;
    б) об’єм оперативної пам’яті не менше 512 Мб;
    в) кількість ядер значно впливає на швидкість алгоритму, тому чим більша кількість ядер, тим швидше будуть виконуватися паралельні обчилення;
- комп’ютерна периферія, до складу якої входить:
    г) монітор;
    д) мишка;
    е) клавіатура.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

5.1 Основні етапи виконання робіт з розробки системи ведення наукової роботи.

№ п/п	Назва етапу роботи	Термін виконання етапу	Результат виконання
1.	Підготовка технічного завдання на розробку програмного продукту	15.02.2019	
2.	Створення архітектури системи	17.02.2019	
3.	Технічне проектування – функціональність, модулі, задачі, цілі тощо	21.02.2019	
4.	Узгодження з керівником інтерфейсу користувача	07.03.2019	
5.	Розробка програми	14.04.2019	
6.	Налагодження програми	22.04.2019	
7.	Тестування програми	16.05.2019	
8.	Здача готового програмного продукту замовнику	27.05.2019	

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

Види, склад, об’єм і методи випробувань програмного продукту повинні бути викладені в програмі і методиці випробувань системи, що розробляється в складі робочої документації.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО

Керівник проекту

_____ *Ю.О. Олійник*
(підпис) (ініціали, прізвище)

“13” травня 2019 р.

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)

“14” травня 2019 р.

Комплекс задач прогнозування часових рядів з використанням технології Apache Spark

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

Шифр *ДП ІС-5220.1181-с.ПМВ*

на 12 сторінках

Київ – 2019 року

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАННЯ

1.1 Найменування програми

Повне найменування програми: Комплекс задач прогнозування часових рядів з використанням технології Apache Spark.

1.2 Область застосування

Функціонал призначений для пошуку прогнозування часових рядів, використовуючи багаторядний метод групового урахування аргументів та технологію Apache Spark.

1.3 Умовне позначення програми

Умовне позначення: «Spark GMDH».

2 МЕТА ВИПРОБУВАНЬ

Метою випробувань є перевірка впливу технології Apache Spark на швидкість, й відновідно якість прогнозу, реалізованого за допомогою багаторядного методу групового урахування аргументів.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Функціональні вимоги:

- створення та редагування вхідних даних прогнозу;
- зміна параметрів моделі для отримання кращих результатів;
- показ часупошуку найкращої моделі;
- отримання графічного звіту з моделювання;
- отримання текстового звіту з моделювання.

3.1.1 Вимоги до складу виконуваних функцій

Функціонал повинен забезпечувати можливість виконання перерахованих нижче функцій:

- створення та редагування простих багаторядних моделей;
- використання базових компонентів для задання поведінки акторів та можливість їх розширення;
- запуск моделювання та отримання звіту;

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація має складатися з керівництва користувача та вихідних текстів програмного коду.

5 СКЛАД І ПОРЯДОК ВИПРОБУВАНЬ

Етапи випробувань:

– ознайомчий;

– виконавчий.

На ознайомчому етапі проводиться:

– перевірка комплектності програмної документації;

– перевірка комплектності складу технічних і програмних засобів.

Під час виконавчого етапу проводиться:

– перевірка відповідності технічних характеристик системи;

– перевірка ступеню виконання вимог функціонального призначення системи.

Функції, що підлягають перевірці:

- створення та редагування моделей;
- запуск алгоритму для вирахування моделей;
- правильність виведення графу на екран відповідно до даних;
- отримання звіту з моделювання.

6 МЕТОДИ ВИПРОБУВАНЬ

У процесі тестування була перевірена основна функціональність засобу для моніторингу та прогнозування фінансових показників. У таблицях 6.1 – 6.5 наведено перелік випробувань основних функціональних можливостей.

Таблиця 6.1 – Ознайомлення з додатком

Дія:	Перегляд допомоги користувача	
	Очікуваний результат	Результат тесту
Передумова		
Відкрити додаток на персональному комп’ютері	Додаток відкритий, користувач знаходиться на головній сторінці	пройдений
Кроки тесту		
Натиснути на клавішу «Допомога»	Відкривається вікно із детальною інформацією про користування програмним забезпеченням	пройдений
Післяумова		
Натиснути на клавішу «Закрити вікно»	Вікно було закрито, користувач повернувся на головне вікно додатку	пройдений

Таблиця 6.2 – Ознайомлення з інформацією про автора

Дія:	Перегляд інформації про автора	
	Очікуваний результат	Результат тесту
Передумова		

Продовження таблиці 6.2

Відкрити додаток на персональному комп'ютері	Додаток відкритий, користувач знаходиться на головній сторінці	пройдений
Кроки тесту		
Натиснути на клавішу «Інформація про автора»	Відкривається вікно із детальною інформацією про автора із посиланням на github	пройдений
Післяумова		
Натиснути на клавішу «Закрити вікно»	Вікно було закрито, користувач повернувся на головне вікно додатку	пройдений

Таблиця 6.3 – Ознайомлення з інформацією про автора

Дія:	Додавання даних до прогнозу	
	Очікуваний результат	Результат тесту
Передумова		
На головному вікні додатку натиснути клавішу «Новий проект»	Відкрите вікно, яке пропонує ввести дані вручну, або зчитати дані з файлу	пройдений
Кроки тесту		
Натиснути на клавішу, що відповідає за зчитування даних з файлу	Відкривається вікно, в якому користувачеві потрібно обрати файл формату *.csv	пройдений
Післяумова		

Продовження таблиці 6.3

Обрати файл та натиснути клавішу «Відкрити»	Дані, що були представлені користувачем з'являються на головному вікні. З'являються параметри, які потрбіно заповнити, щоб запустити модель	пройдений

Таблиця 6.4 – Ознайомлення з інформацією про автора

Дія:	Додавання даних до прогнозу	
	Очікуваний результат	Результат тесту
Передумова		
Відкрити дані з жорсткого диску, або ввести дані вручну	Дані з'являються на головному вікні додатку. З'язлаються поля, в які потбірно ввести параметри пошуку моделей	пройдений
Кроки тесту		
Заповнити всі представлені параметри та натиснути клавішу «Run»	Алгоритм починає вираховувати найкращу модель. Після обрахунку спливає вікно, на якому зазначено скільки часу алгоритм рахував.	пройдений
Післяумова		
Закрити вікно, що показує час прорахунку алгоритму.	Справа на головному вінкі додатку з'являються опції як користувач може переглянути отримані дані прогнозу	пройдений

Таблиця 6.5 – Ознайомлення з інформацією про автора

Дія:	Додавання даних до прогнозу	

Продовження таблиці 6.5

	Очікуваний результат	Результат тесту
Передумова		
Закрити вікно, що показує час прорахунку алгоритму	Справа на головному вінкі додатку з’являються опції як користувач може переглянути отримані дані прогнозу	пройдений
Кроки тесту		
Справа натиснути на клавішу «Data»	З’являється вікно, в якому показана головна модель, що була прорахована алгоритмом, критерій моделі та також представлені моделі для алргументів	пройдений
Післяумова		
Закрити вікно із даними	Користувач повертається на головне вікно додатку	пройдений

Таблиця 6.6 – Ознайомлення з інформацією про автора

Дія:	Додавання даних до прогнозу	
	Очікуваний результат	Результат тесту
Передумова		
Закрити вікно, що показує час прорахунку алгоритму	Справа на головному вінкі додатку з’являються опції як користувач може переглянути отримані дані прогнозу	пройдений
Кроки тесту		
Справа натиснути на	З’являється вікно, в якому представлений графік із поточними даними та	пройдений

клавішу «Line»	із даними прогнозу	

Продовження таблиці 6.6

Післяумова		
Закрити вікно з графіком	Користувач повертається на головне вікно додатку	пройдений

Графічний матеріал
до дипломного проекту

на тему: Комплекс задач прогнозування часових рядів за допомогою технології Apache Spark

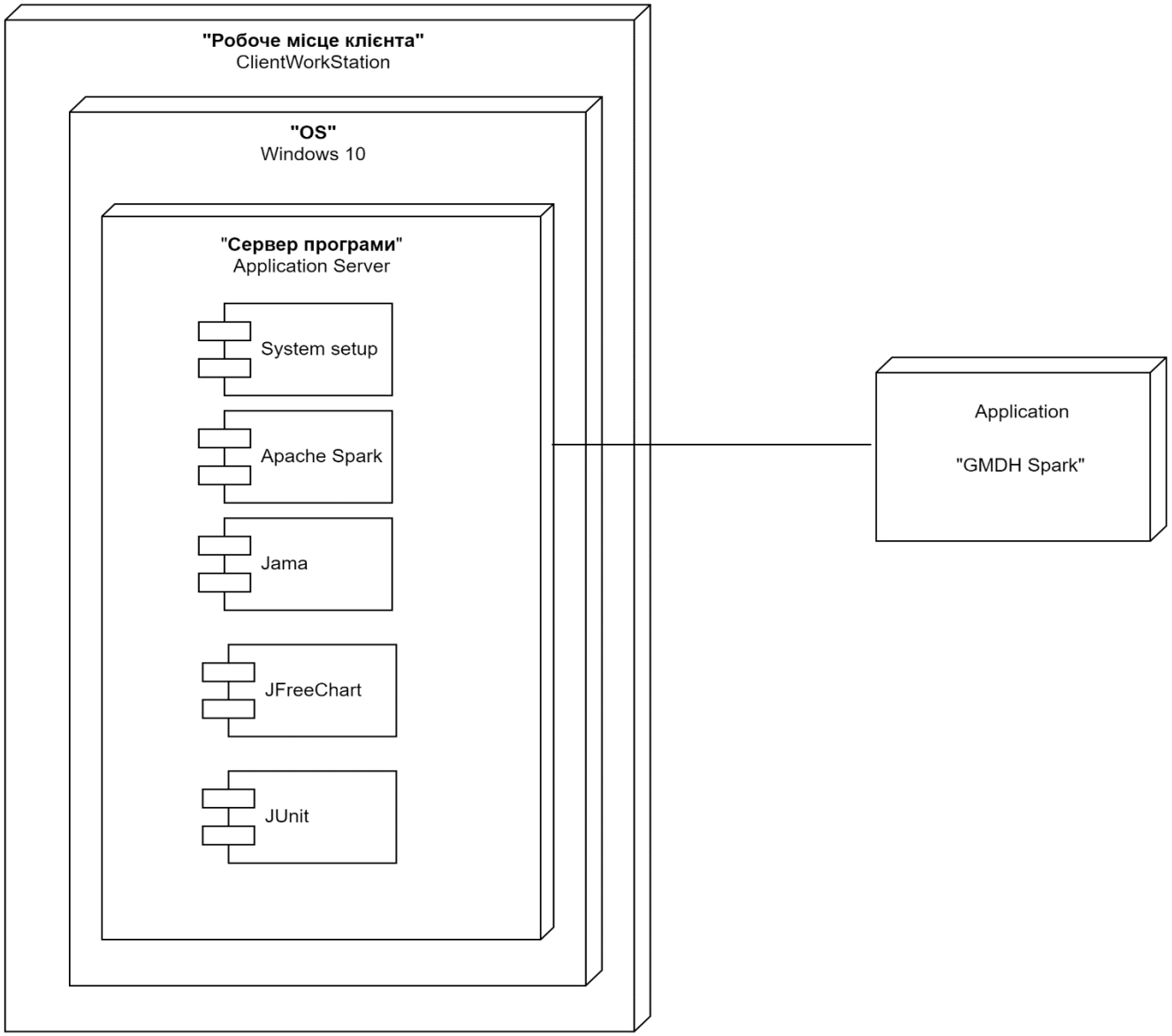
Київ – 2019 року

					ДП ІС-5220.1181-с.ССД		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна діяльності	Літера	Маса / Масштаб
Розробив		Олійник Я.М.					
Перевірив		Олійник Ю.О.				Аркуш 1	Аркушів 1
Т. кон.					Комплекс задач прогнозування часових рядів із використанням технології Apache Spark	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н. кон.		Халус О.А.					
Затвердив		Олійник Ю.О.					

ДП ІС-5220.1181-с.ССД

					ДП ІС-5220.1181-с.ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Олійник Я.М.				
Перевірив		Олійник Ю.О.				Аркуш 1 / Аркушів 1
Т. кон.					Комплекс задач прогнозування часових рядів з використанням технології Apache Spark	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Н. кон.		Халус О.А.				
Затвердив		Олійник Ю.О.				

ДП ІС-5220.1181-с.ССД



					ДП ІС-5220.1181-с.ССК			
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема стректурна компонентів програмного забезпечення			
Розробив		Олійник Я. М.				Аркуш 1	Аркушів 1	
Перевірив		Олійник Ю.О.						
Т. кон.					Комплекс задач з прогнозування часових рядів з використанням технології Apache Spark	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52		
Н. кон.		Халус О.А.						
Затвердив		Олійник Ю.О.						

ДП ІС-5220.1181-с.ССП

					ДП ІС-5220.1181-с.ССП		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна послідовності	Літера	Маса / Масштаб
Розробив		Олійник Я.М.					
						Аркуш 1	Аркушів 1
Перевірив		Олійник Ю.О.					
Т. кон.					Комплекс задач прогнозування часових рядів із використанням технології Apache Spark	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н. кон.		Халус О.А.					
Затвердив		Олійник Ю.О.					

ДП ІС-5220.1181-с.ССД

					ДП ІС-5220.1181-с.ССД			
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна варіантів використань*	Літера	Маса	Масштаб
Розробив		Олійник Я.М.						
						Аркуш 1		Аркушів 1
Перевірив		Олійник Ю.О.						
Т. кон.					*Комплекс задач прогнозування часових рядів з використанням технології Apache Spark*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52*		
Н. кон.		Халус О.А.						
Затвердив		Олійник Я.М.						